Universidad
Carlos III de Madrid

PROYECTO FIN DE CARRERA

DISEÑO E IMPLEMENTACIÓN DE UN PLUGIN PARA UN CLIENTE BITTORRENT MEJORADO CON UN SISTEMA DE RECOMENDACIÓN Y DETECCIÓN DE CONTENIDOS FALSOS

Autora: **Beatriz Calvo González**

Tutor: **Rubén Cuevas Rumín**

Junio 2011

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. RESUMEN

Las redes P2P (*Peer-to-Peer*) [1] [2] son la herramienta más utilizada para la compartición de ficheros. Su aparición conllevó el crecimiento del número de usuarios de Internet. La popularidad de estas redes de intercambio se mantiene debido a la posibilidad de disponer de contenidos como música, software o películas sin coste para los usuarios de manera sencilla.

La disponibilidad de contenidos en *streaming* en Internet, la incorporación de sistemas a la carta en televisión o los servidores de contenido son algunos de los agentes que compiten contra las redes P2P pero éstas siguen siendo muy utilizadas.

El protocolo BitTorrent sigue la filosofía P2P y es el sistema de intercambio de contenido más utilizado. Sólo teniendo en cuenta dos de los programas que implementan este protocolo, *BitTorrent Mainline* y *µTorrent*, se llega a más de 400000 descargas nuevas al día de los clientes y 20 millones de usuarios diarios. [3]

Sin embargo, el protocolo BitTorrent presenta la desventaja de estar muy ligado a las acciones que realice el usuario, como escoger sus descargas o localizar los ficheros *torrent*. Por este motivo las aplicaciones P2P están apostando por la descentralización y las recomendaciones de contenidos.

El enfoque centralizado de BitTorrent viene dado por [4]:

- Necesita un servidor que coordine la descarga de contenido.
- Requiere un archivo *torrent* para iniciar la descarga, normalmente disponible a través de páginas que mantienen un índice de contenidos, aunque puede ser publicado y distribuido a través de otros métodos.
- No ofrece al usuario ninguna garantía o método sencillo de comprobación del contenido compartido en un *torrent*, por ello las páginas de descargas o foros de intercambio actúan como filtros de recomendación y control de calidad evitando así contenidos falsos, inadecuados o peligrosos.

Por este motivo, en la actualidad se pretende dotar a las aplicaciones P2P existentes de herramientas que eviten estas dependencias. Aunque están apareciendo también nuevos clientes P2P adaptados a las nuevos requerimientos pero que aún no disponen de la popularidad de los ya existentes.

En concreto, este Proyecto está enfocado a automatizar el procedimiento llevado a cabo por los usuarios del cliente BitTorrent llamado *Vuze* (pudiéndose extender en el futuro a otros clientes) basándose en las siguientes líneas de desarrollo:

- Almacenar información detallada de las descargas llevadas a cabo por los usuarios para poder desarrollar un sistema de recomendación basado en los historiales de descargas.

- Proveer de un sistema de alerta que avise al usuario de la falsedad del contenido de sus descargas basándose en los resultados aportados por una aplicación que detecta fakes.

Vuze ha sido el cliente escogido para el desarrollo de este Proyecto ya que cuenta con cerca de 100 millones de usuarios al mes y es el cliente elegido por el 25% de los usuarios de *The Pirate Bay* [5]. Por ello, la aplicación cliente-servidor desarrollada para este cliente pretende aprovechar su popularidad solventando las necesidades actuales del gran número de usuarios que dispone y facilitando la utilización del programa.

1.2. OBJETIVOS

Este Proyecto Fin de Carrera se enmarca en un trabajo más extenso llevado a cabo por el *Departamento de Ingeniería Telemática* cuyo objetivo es mejorar las prestaciones de los clientes BitTorrent actuales.

En concreto este Proyecto está orientado al desarrollo de un sistema de recomendación de contenidos y detección de falsedad en los mismos. Por lo que se tiene como finalidad implementar un plugin para los usuarios de *Vuze* que deseen ampliar la funcionalidad de su cliente BitTorrent con las mejoras descritas anteriormente.

De este modo, se dotará al popular cliente *Vuze* de un sistema adaptado a los actuales requerimientos de los usuarios. Quedará automatizado en cierto modo el procedimiento llevado a cabo por los usuarios para iniciar sus descargas facilitándole contenidos que puedan interesarle y, a su vez, advirtiéndole de la falsedad de los archivos que no correspondan con lo esperado.

En concreto, este Proyecto recoge información de los usuarios de *Vuze* y de sus descargas a partir de la cual se desarrollará el sistema de recomendación de contenidos basándose en coincidencias entre usuarios. Por lo tanto, el objetivo es el desarrollo de una herramienta software encargada de reunir información de las descargas de los usuarios del programa *Vuze* que instalen el plugin.

Aparte de almacenar la información de las descargas, la aplicación deberá mantener todos los datos actualizados para que el sistema se adapte completamente al contexto de cada usuario permitiendo una recomendación acertada en todo momento.

En la aplicación participarán varias entidades con los siguientes objetivos:

- Plugin: Es el software a instalar en *Vuze* por los usuarios interesados. Se encargará de recolectar parámetros de las descargas, como el nombre del fichero o el identificador del *torrent* asociado a una descarga, y de su envío a un servidor.

- Servidor: Mientras que se ejecute este módulo los clientes que tengan instalado al plugin se encargarán de remitirle la información concerniente a sus descargas. El servidor almacenará en una base de datos el listado de usuarios y sus descargas.

- Base de datos: Este módulo almacenará la información recogida por el plugin de manera estructurada facilitando que el acceso a los datos para el desarrollo del sistema de recomendación sea sencillo.

Por otro lado, el Proyecto cuenta con otro objetivo que consiste en desarrollar un sistema de alerta que advierta al usuario de la correspondencia del contenido de sus descargas con lo que espera de ellas. Para ello se recurrirá a resultados reales de la aplicación *FakeTorrent*.

1.3. TRABAJOS RELACIONADOS

1.3.1. Sistemas de recomendación

Dada la dificultad de parametrizar de forma automática algunos contenidos, surgió la idea de desarrollar sistemas de recomendación [6] [7] cuya idea es encontrar usuarios con gustos similares a los de otro determinado y recomendar a éste contenidos que desconoce pero que interesan a aquéllos con los que tiene similitud. Se consigue así una inteligencia colectiva que permite al usuario recibir información adaptada a sus gustos.

El desarrollo de este Proyecto está orientado a la futura elaboración de un sistema de recomendación de contenidos para *Vuze*. Desarrollar un sistema de este tipo en un cliente BitTorrent mejora en gran medida las prestaciones de éste y las facilidades aportadas al usuario.

Para ello se recurre al filtrado colaborativo que se basa en las preferencias de otros usuarios que manifiestan gustos parecidos. De esta manera, se propone al usuario qué archivos pueden interesarle en función de los historiales de descargas recogidos por la aplicación. El algoritmo de filtrado colaborativo que consigue este objetivo es el basado en memoria, el cual recurre a una base de datos de elementos y usuarios para generar predicciones. Primero se emplean técnicas que localicen los usuarios con un historial similar al del usuario actual. A partir de la combinación de preferencias se generan los elementos más recomendables para el usuario actual.

En la actualidad, se utilizan sistemas de recomendación en un gran número de sitios web, buscadores, redes sociales, etc. Por ejemplo, el portal *Amazon.com* utiliza un sistema de recomendaciones para asesorar de forma personalizada en la compra de productos. La radio en Internet *Last.Fm* incluye un programa de recomendación en base a la música escogida. El buscador *Google* cuenta con un sistema de recomendación de páginas similares a la buscada (*Google Similar pages*).

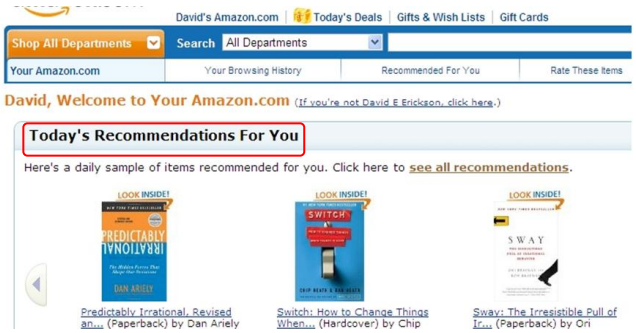


Figura 1.1. Captura con recomendaciones hechas en Amazon

En este Proyecto se facilita la base para desarrollar un sistema de recomendación en *Vuze* que conseguirá reducir la dependencia que presenta el protocolo BitTorrent a las acciones que realice el usuario. Se proporcionarán recomendaciones de contenidos en base a la comparación del comportamiento previo de usuarios con gustos similares.

1.3.2. Detección de contenidos falsos

El protocolo BitTorrent no ofrece al usuario ninguna garantía sobre el contenido que va a descargar asociado a un archivo torrent. El título del fichero puede no corresponderse al contenido del mismo (este hecho es conocido como "*fake*"") por lo que, en ese caso, el cliente descarga un archivo que no coincide con el esperado.

Averiguar la veracidad del contenido de una descarga puede realizarse en función a varios parámetros que dan pistas para determinarlo. Los comentarios de otros usuarios a la descarga, el tamaño del fichero, los marcadores de advertencia y de verificación o la calificación que tiene el usuario que comparte el archivo son algunos de los parámetros que pueden llevar a decidir si un contenido es o no falso.

Existen páginas web que a partir del nombre del torrent o del identificador (*infohash*) informan si el contenido asociado es falso. Entre ellas destaca *Vertor* (VERified TORrents) que es una base de datos en línea de torrents que han sido comprobados para evitar la descarga de contenidos falsos [8].

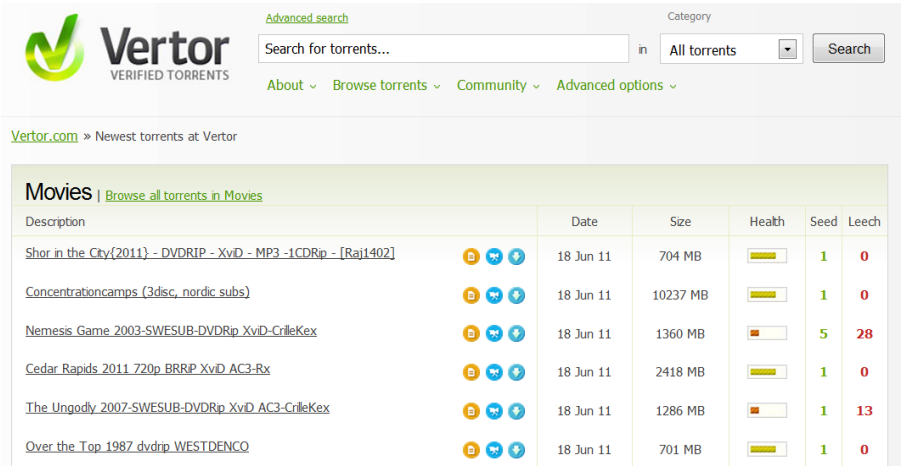


Figura 1.2. Interfaz de la página web Vertor

Según datos recientes [9] se sabe que el 30% de los contenidos que circulan por la red BitTorrent son distribuidos por agencias antipiratería sufragadas por productoras musicales, cinematográficas y desarrolladores de software así como usuarios maliciosos que infectan los archivos con virus o troyanos. La motivación es contaminar los *trackers* con versiones falsas del contenido que quieren proteger. A partir de estos datos se obtiene un patrón que caracteriza a los distribuidores de contenidos falsos a partir de sus comportamientos. El análisis de dicha información pretende evitar que los usuarios lleven a cabo este tipo de descargas avisándoles cuando vaya a ocurrir una de ellas.

En el *Departamento de Ingeniería Telemática* se ha desarrollado una aplicación que determina la correspondencia de un contenido con lo que realmente se espera en él. Esta

decisión se toma a partir del infohash que caracteriza a cada torrent. Se cuenta con una base de datos donde se tienen controlados los torrent identificando si su contenido es falso o no.

En este Proyecto se ha integrado el plugin desarrollado con la aplicación descrita anteriormente ofreciendo datos reales de la autenticidad de un contenido, sin necesidad de consultarlo de manera externa a *Vuze,* y avisando con alertas informativas al usuario.

1.4. ESTRUCTURA DE LA MEMORIA

La memoria del Proyecto está dividida en siete capítulos de los que a continuación se muestra un breve resumen:

- **Introducción.** Contiene un breve introducción a la situación de las redes P2P y, en concreto, del protocolo BitTorrent. Se plantean los objetivos de este Proyecto y trabajos relacionados con los mismos. Por otro lado se trata la metodología de trabajo que se ha seguido enunciando las distintas fases.
- **Protocolo BitTorrent.** Trata sobre el funcionamiento de este protocolo, sus características, las entidades que participan en la compartición y los archivos que requiere.
- **Clientes BitTorrent.** Este apartado describe los clientes BitTorrent más importantes en la actualidad con sus principales características detallando en mayor medida las del cliente *Vuze*. También trata de los últimos clientes BitTorrent desarrollados.
- **Aplicación desarrollada**. Se realiza una breve introducción a la aplicación desarrollada en este Proyecto pasando a exponer con detalle la funcionalidad y características de cada uno de los módulos que la forman, la comunicación entre éstos, la descripción de los sistemas desarrollados y el funcionamiento de la aplicación junto a un ejemplo. Por último, se indica la estructura de ficheros que conforma la aplicación.
- **Herramientas.** Incluye una descripción de las herramientas y tecnologías empleadas en la implementación de la aplicación.
- **Conclusiones y trabajos futuros.** Este apartado presenta las conclusiones extraídas de la realización de este Proyecto. Además, se citan algunos aspectos que ampliarían la aplicación y posibles trabajos futuros.
- **Anexos.** Por último se adjuntan tres anexos: El primero es el presupuesto orientativo que supone la realización de este Proyecto con los costes aproximados (*ANEXO 1. Presupuesto*), el segundo trata los pasos a seguir para arrancar el programa servidos en una máquina (*ANEXO 2. Manual de puesta en marcha del servidor*) y el tercero contiene las indicaciones a seguir por los usuarios que instalen el plugin (*ANEXO 3. Manual del plugin para el usuario*).

1.5. METODOLOGÍA DE TRABAJO

Para realizar este Proyecto se ha seguido la siguiente metodología de trabajo:

- Definición de los objetivos iniciales para definir la orientación del Proyecto.
- Toma de contacto con el protocolo BitTorrent y los diversos programas clientes, profundizando en *Vuze*.
- Instalación de las herramientas necesarias para poder desarrollar la aplicación.
- Seguimiento de la guía de desarrollo de plugins para *Vuze*.
- Estudio de las alternativas de implementación y elección de las tecnologías y herramientas involucradas.
- Implementación del programa que se instala en el cliente responsable de recoger los datos y enviarlos al servidor.
- Diseño y creación de la base de datos que almacena la información de los usuarios y sus descargas.
- Implementación del programa servidor y la comunicación con la base de datos.
- Integración con la aplicación que detecta contenidos falsos.
- Verificaciones y pruebas a la aplicación.
- Generación de los ficheros necesarios para poder portar la aplicación.
- Análisis de los resultados y planteamiento de posibles mejoras y líneas de trabajo futuras.
- Elaboración de la memoria.

2. PROTOCOLO BITTORRENT

2.1. INTRODUCCIÓN

La posibilidad de descargar contenidos implica que alguien debe disponer de ellos y compartirlos. Bajo este escenario se encuentran las redes P2P (Peer-To-Peer) que han supuesto un avance en este ámbito aprovechando el ancho de banda de los usuarios que ya cuentan con el contenido para que éstos colaboren en la difusión de los archivos.

BitTorrent [10] [11] [12] [13] es un protocolo de software libre de compartición de archivos P2P (*Peer-to-Peer*) diseñado por Bram Cohen en el año 2001. Su idea fundamental es que cada usuario funciona simultáneamente como cliente y servidor para evitar que un servidor central se encargue de enviar el contenido, sino que sólo deba controlar el estado de la red. Por lo tanto, el principal objetivo de BitTorrent es proporcionar de manera eficiente la distribución de un mismo fichero a un gran grupo de personas, forzando a todos los que descargan un fichero a compartirlo también con otros.

Según esto, el protocolo BitTorrent ha demostrado ser una solución óptima al problema de congestión en Internet demostrando una alta capacidad de escalabilidad y robustez en el servicio.

2.2. FUNCIONAMIENTO

Según el protocolo BitTorrent primero se distribuye un fichero de tamaño reducido con extensión *.torrent*. Se trata de un fichero estático que se puede encontrar en páginas web o distribuirse incluso por correo electrónico. Este fichero contiene la dirección de un servidor de búsqueda que se encarga de localizar fuentes con el fichero completo o con partes de él.

Dicho servidor se encuentra centralizado y provee de estadísticas acerca del número de transferencias, el número de nodos que disponen de una copia completa del fichero y el número de nodos que poseen sólo una porción.

A partir de las fuentes localizadas por el servidor de búsqueda se procede a descargar el fichero deseado. Al mismo tiempo que se realiza la descarga, se comienza a subir las partes disponibles a otras fuentes en base al ancho de banda asignado para ello. Según esto, cada nodo contribuye a la distribución del fichero al comenzar a compartirlo antes de completar la descarga.

Cuando un usuario comienza la descarga de un fichero, no se empieza necesariamente por el principio del mismo, sino que se descarga por partes desordenadas. Si entre los usuarios conectados se dispone de la totalidad de partes que forman el fichero completo, aunque éstas se encuentran dispersas, todos obtendrán una copia completa de él. Inicialmente algún nodo debe poseer el fichero completo para que el proceso pueda comenzar.

Este mecanismo conlleva que cuando no existan nodos con el fichero completo (semillas o seeds) conectados al servidor de búsqueda, es posible que el fichero no pueda ser completado.

2.3. ESTRUCTURA DE LA RED

En la distribución de ficheros según BitTorrent participan las siguientes entidades:

- **Tracker**: (en el centro de la *Figura 2.1*) Es un servidor central que conoce quiénes comparten cada fichero. Conoce dónde se encuentran todos los usuarios y se encarga de gestionarlos asignando a cada cliente "n" posibles destinatarios aleatorios para realizar la descarga.
- **Seeds:** (ordenadores de color azul en la *Figura 2.1*). Son los usuarios que disponen de la totalidad del archivo que se quiere descargar. Se encargan de enviarlo al resto de usuarios.
- **Peers:** (ordenadores de color negro en la *Figura 2.1*). Son el resto de usuarios que se dedican a descargar los archivos de todos los peers y, a la vez, suben el fragmento que tienen de archivo a otros peers.
- **Leechers:** Son todos los usuarios que están en la red descargando un archivo pero que todavía no tienen el archivo completo.

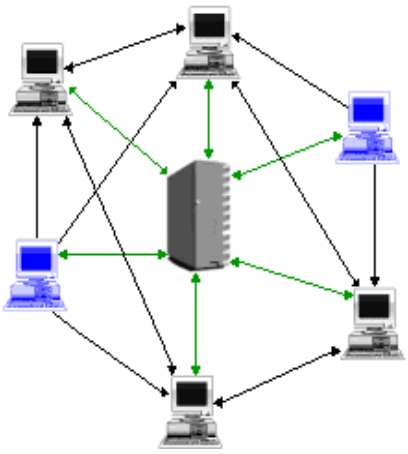


Figura 2.1. Esquema de la estructura de una red BitTorrent

2.4. MECÁNICA DE DESCARGAS

El proceso de descarga de un fichero según el protocolo BitTorrent es el siguiente:

1. Un usuario baja de un servidor web el archivo con extensión *.torrent* que contiene la dirección del tracker al que hay que conectarse para pertenecer a la red de peers.
2. Con un programa cliente como *Vuze*, el cual es empleado en el desarrollo de este Proyecto, se abre el archivo *.torrent*.

3. Entre el tracker y el peer existe una comunicación a través de una conexión HTTP. El tracker informa del listado de peers y seeds que contienen partes del archivo que se quiere descargar y lo actualiza con la información del nuevo peer.
4. El peer conoce dónde localizar las partes del archivo por lo que se comunica con otros mediante sockets TCP o UDP. De este modo la descarga del archivo comienza en el ordenador del usuario. Ahí comienza la compartición automática con otros peers.

2.5. ARCHIVOS .TORRENT

Los archivos torrent tienen un papel de gran importancia en el protocolo BitTorrent debido a que son los encargados de enlazar la descarga con el tracker y con el resto de usuarios que están descargando el archivo.

Están codificados mediante un formato llamado "bencoding". Son usados por el tracker y el cliente.

Los archivos torrent son publicados en la web y, a su vez, son registrados en el tracker que mantiene la lista de clientes que actualmente participan sobre el archivo torrent.

El contenido de estos archivos es un diccionario que contiene las siguientes claves:

- **info**: es un diccionario que describe los archivos del torrent con diferente estructura dependiendo de si el torrent es para bajar un archivo o varios. Las claves que contiene son:
    - **name**: Cadena con el nombre del archivo o directorio donde se almacenarán los archivos.
    - **piece length**: Entero que representa el número de bytes de cada pieza.
    - **pieces**: Cadena que representa la concatenación de la lista de claves hash de cada parte del fichero compartido para asegurar la integridad de cada parte completada.
    - **private**: Clave opcional formada por un entero de valor 0 ó 1 que indica si se pueden buscar peers fuera de los trackers descritos en la metainformación.
    - **length**: Entero con la longitud del archivo en bytes.
    - **md5sum**: Cadena opcional de 32 caracteres en hexadecimal correspondiente a la suma MD5 del archivo.
    - **files**: Es una lista de diccionarios (uno por cada archivo) que existirá en el caso de que sea un torrent de múltiples archivos.
- **announce**: cadena que representa la URL del servidor central.
- **announce-list**: (lista de cadenas opcional). Representar listas de trackers alternativos.
- **creation date**: (entero opcional) La fecha de creación del torrent en formato de época UNIX.
- **comment**: (cadena opcional) Campo libre para el creador del torrent.

- **created by**: (cadena opcional) Nombre y versión del programa usado para crear el archivo.

Estos archivos se encuentran alojados en motores de búsqueda que ofrecen la posibilidad de descargar los torrents. Algunos ejemplos son:

- The Pirate Bay http://thepiratebay.org
- Mininova http://www.mininova.org
- Torrentz http://torrentz.eu
- The Torrent Bay http://thetorrentbay.blogspot.com
- Torrents.net http://www.torrents.net

2.6. CARACTERÍSTICAS

- Los archivos son divididos en pequeñas porciones que son compartidas bajo la premisa: "la porción menos disponible primero".
- En una red P2P descentralizada cada usuario conectado a la red comparte los archivos que dispone y a la vez tiene acceso a los que poseen el resto de los usuarios.
- El hecho de que múltiples usuarios puedan actuar como servidores de contenido permite una mayor robustez en el servicio. En el supuesto de que un usuario se desconectara, puede haber otros desde los cuales conseguir el contenido deseado.
- La velocidad de transferencia se ve optimizada cuando muchos usuarios se conectan para bajar un mismo fichero.
- La desaparición de archivos se lleva a cabo en el momento que el tracker los descarta y éste suele hacerlo a un tiempo determinado sin conexión de ningún seed en función de la configuración del tracker.

3. CLIENTES BITTORRENT

3.1. INTRODUCCIÓN

Un cliente BitTorrent es una aplicación que permite a un usuario establecer una conexión tipo P2P (*Peer-To-Peer*) para descargar archivos que otros usuarios poseen y quieren compartir facilitando el intercambio de los mismos.

Existen dos tipos de clientes BitTorrent:

- **De múltiples descargas simultáneas**
  - Ejemplos: *Vuze*, *BitComet*, *KTorrent*, *µTorrent* o *Transmission*.
- **De descarga única**
  - Sólo descarga un archivo torrent, pero se pueden tener varios abiertos simultáneamente.
  - Ejemplos: *BitTornado* o *navegador Opera*.

Son muchos los clientes BitTorrent disponibles para descargar y compartir archivos: *BitTorrent Oficial*, *BitTornado*, *BitComet*, *Tribler*, *Vuze*, *Deluge*, etc.

A continuación se describen los cinco clientes BitTorrent que según un estudio realizado por la Universidad Tecnológica de Delft (Holanda) son los más utilizados.[14]

Se comienza por *Vuze* profundizando en aspectos como la posibilidad de instalación de plugins o su funcionamiento dado que ha sido el programa escogido para el desarrollado del plugin. Posteriormente se detallan características generales de los clientes: *BitTorrent Oficial*, *µTorrent*, *BitComet* y *Transmission*.

Figura 3.1. Logotipos de los clientes BitTorrent más utilizados

Por último, se trata la aparición de clientes de reciente desarrollo pero que incluyen funcionalidades de gran interés para el desarrollo de estas tecnologías: *Tribler*, *Chrysalis* y *BitMate*.

Figura 3.2. Logotipos de tres nuevos clientes BitTorrent

3.2. VUZE www.vuze.com

3.2.1. Descripción

El cliente BitTorrent *Vuze* [15] [16] [17] era conocido anteriormente como *Azureus*. La creación de la plataforma de distribución de contenidos *Vuze* fue en 2007 de mano del equipo de desarrollo de *Azureus* que con el tiempo pasó a ser sustituido completamente por *Vuze*. Este cliente está desarrollado en lenguaje Java y es de código abierto lo que permite que se desarrollen aplicaciones que ofrecen nuevas características de manera más accesible. Tiene licencia GNU GPL lo que lo protege en la libre distribución, modificación y uso de su software.

Vuze es multiplataforma por lo que está disponible para sistemas operativos como Microsoft Windows, MAC OS o Linux. Necesita tener instalado en la máquina donde se utilice el JRE (*Java Runtime Environment*).

Vuze permite que los usuarios establezcan conexiones cuyo fin es descargar e intercambiar contenidos. Además, añade un componente social que permite a los usuarios, por ejemplo, añadir amigos al cliente y priorizar el tráfico para compartir más fácilmente los archivos con sus amigos.

3.2.2. Interfaz

Vuze cuenta con una interfaz gráfica de usuario (*GUI*) desde la cual se permite al usuario descargar múltiples archivos simultáneamente. También ofrece gran cantidad de ajustes de usuario configurables y estadísticas como:

- Velocidades actuales de descarga y subida.
- Tiempo estimado restante para completar una descarga.
- Detalles sobre cada parte del archivo a descargar como el número o la disponibilidad de cada parte.
- Nombres de los archivos y tamaño de los mismos.
- Datos del peer como las direcciones IP, las velocidades con las cuales se está descargando y subiendo desde/hacia él, el puerto con el que se encuentra funcionando en BitTorrent, y el cliente de BitTorrent que él está utilizando.

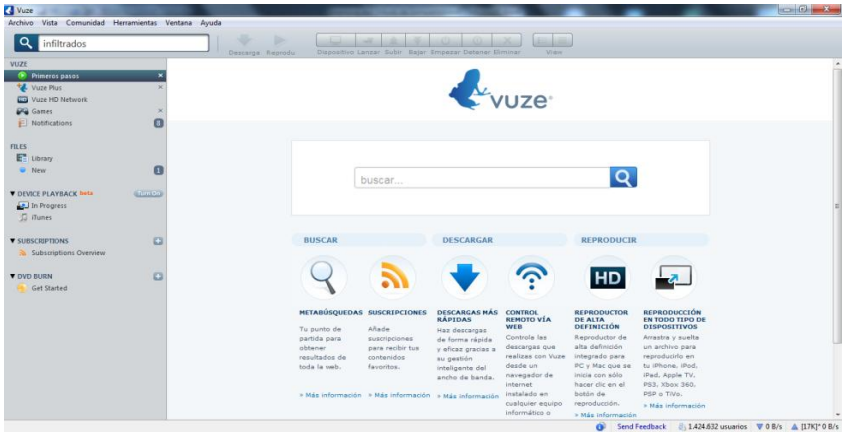


Figura 3.3. Interfaz de Vuze empleada en el desarrollo de la aplicación.

3.2.3. Modos de funcionamiento

Para iniciar una descarga *Vuze* ofrece dos posibilidades:

→ *Abrir un archivo torrent*

Dicho fichero se encuentra guardado por el usuario que con anterioridad lo ha obtenido, generalmente, de una página especializada.

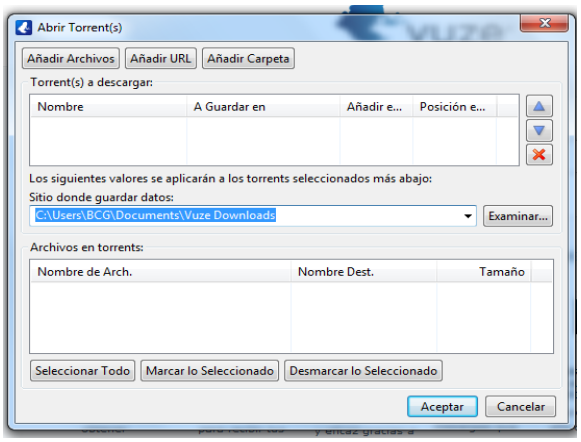


Figura 3.4. Apertura de un fichero torrent desde Vuze

→ *Recurrir al metabuscador*

Vuze simplifica la búsqueda de archivos para el usuario con un cuadro de búsqueda como punto de partida. Se evita así acceder a diversos sitios para localizar el contenido deseado. Hace uso de distintos buscadores y recopilatorios de enlaces torrent como *Mininova* para localizar resultados. *Vuze* da la opción de añadir más motores de búsqueda además de los ya listados en el programa.

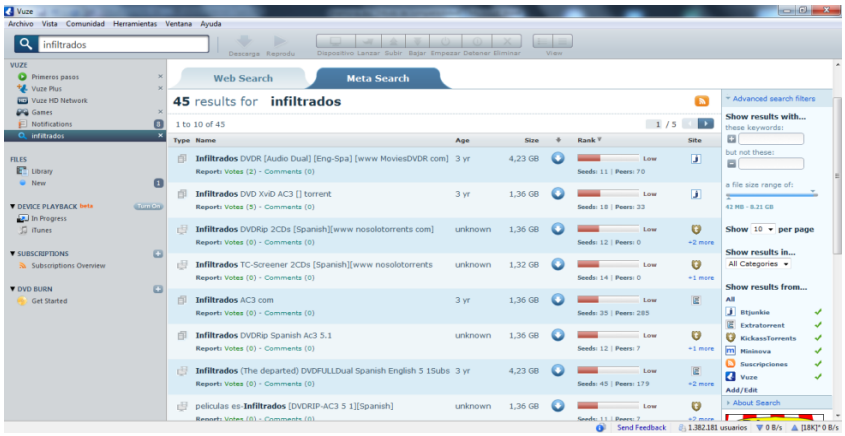


Figura 3.5. Interfaz de Vuze en la búsqueda de un archivo desde el buscador

A partir de disponer del fichero torrent, *Vuze* sigue la mecánica de funcionamiento del protocolo BitTorrent para realizar las descarga del mismo (detallada en el apartado *2.4. MECÁNICA DE DESCARGAS*) a través de redes de pares con lo que se consigue que los usuarios intercambien sus contenidos, le asignen categorías, comentarios y calificaciones.

3.2.4. Características generales [18]

- Permite que los usuarios configuren un límite de velocidad en la subida y la descarga.
- Ofrece un canal con contenido multimedia en alta definición o calidad DVD llamado *Vuze HD Network* gracias un servicio de contenidos de la compañía *Vuze Inc*.
- Da la opción de previsualizar algunos archivos que se han descargado antes de que la descarga haya finalizado para evitar contenidos falsos.
- Puede servir como un tracker propio permitiendo así compartir sus propios archivos sin subirlos a otro lugar.
- Soporta el cifrado *Message Stream Encryption* para evitar mensajes de spam.
- Soporta el intercambio de fuentes entre usuarios denominado Peer exchange.
- Permite ampliar la funcionalidad con mods o plugins de los cuales se detallan en el siguiente apartado.

3.2.5. Plugins

La funcionalidad que ofrece *Vuze* se puede ampliar con complementos (*plugins*) [16] que se desarrollan para mejorar sus prestaciones tras ser instalados desde el programa. *Vuze* proporciona una interfaz de programación de aplicaciones (*API*) que facilita el desarrollo de estos plugins.

Los plugins pueden ser instalados y desinstalados a voluntad del usuario desde el programa y son desarrollados con posterioridad al programa principal.

Entre las funcionalidades que pueden aportar los plugins se encuentran:

- Ajuste de las velocidades de carga y descarga.
- Programación de límites de velocidad de carga y descarga.
- Auto-descarga de torrents con RSS Feed Scanner que es un analizador automático altamente configurable que permite la descarga automatizada de torrents a través de sus capacidades avanzadas de filtrado.
- Recepción de un correo informando del fin de una descarga con Status Mailer.
- Ampliación de la información en torrents como la localización de país.
- Control remoto de *Vuze* a través de las interfaces Web Swing o HTML Web.
- Integración con la red de Nodezilla Grid Network permitiendo el almacenaje, compartición y publicación anónima de archivos torrent.

Existe un listado de plugins disponibles agrupados por las siguientes categorías: Core, Automation, More Info & User-friendliness, Remote Access, Networks, Games y Developer Samples. En él se muestra una breve descripción del complemento y se indica la última versión disponible junto al enlace para descargarlo. Este listado se encuentra en la siguiente dirección: *http://azureus.sourceforge.net/plugin_list.php*

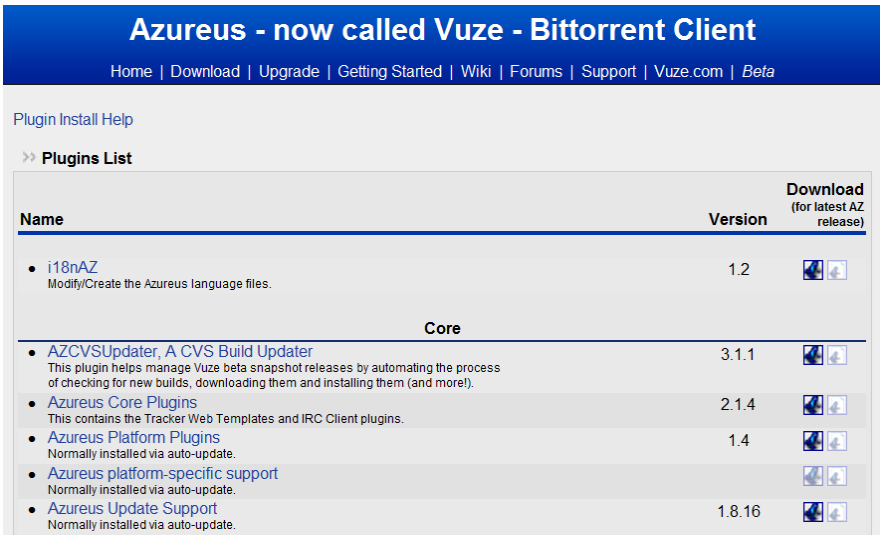


Figura 3.6. Página con el listado de plugins disponibles para Vuze

Vuze ofrece también una guía con las instrucciones para la instalación de plugins accesible en: *http://wiki.vuze.com/w/Install_Plugins*

3.3. µTORRENT www.utorrent.com

Actualmente *µTorrent* [19] [20] es el cliente BitTorrent más popular debido a su velocidad en las descargas y a los pocos recursos que consume. Está escrito en C++ y funciona en Windows y Mac OS. Para utilizarlo en Linux es necesario se emplea la capa de compatibilidad Wine. No es una aplicación de código abierto.

Entre sus características principales se encuentran:

- Descarga de múltiples ficheros desde múltiples seeds al mismo tiempo.
- Alta velocidad de descarga.
- Consumo de muy pocos recursos.
- Selección de los archivos que se quieren bajar dentro de un *torrent*.
- Compatibilidad con el estado de hibernación en Windows.
- Apagado automático al acabar las descargas.
- Planificador del ancho de banda en función de las horas del día.
- Posibilidad de añadir la capacidad de servidor web, pudiendo manejar remotamente las descargas y su configuración.
- Suscripción a canales RSS para descargar automáticamente contenido especializado del mismo.

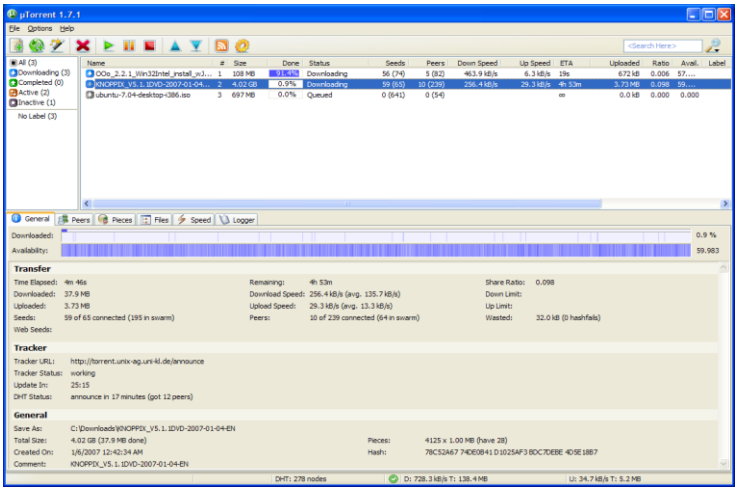


Figura 3.7. Interfaz del cliente µTorrent

3.4. BITTORRENT OFICIAL www.bittorrent.com

Fue el primer programa escrito para el protocolo BitTorrent. Para distinguirlo de otros clientes y evitar confusiones con el nombre del protocolo se denomina *BitTorrent Oficial* o *BitTorrent Mainline*.

Fue desarrollado originariamente por el programador Bram Cohen. Desde el lanzamiento de la versión 6.0 tiene las siguientes características: dejó de ser código abierto, está escrito en C++ y sólo funciona bajo Windows.

Es muy similar al *µTorrent* ya que este cliente fue adquirido por *BitTorrent, Inc.* aunque ambos siguen desarrollándose de forma independiente.

Este cliente BitTorrent permite buscar y descargar archivos *torrent* usando una caja de búsqueda incluida en la ventana principal, el cual abre el sistema de búsqueda de BitTorrent con los resultados de la búsqueda en el navegador web predeterminado del usuario final. [10] [11]

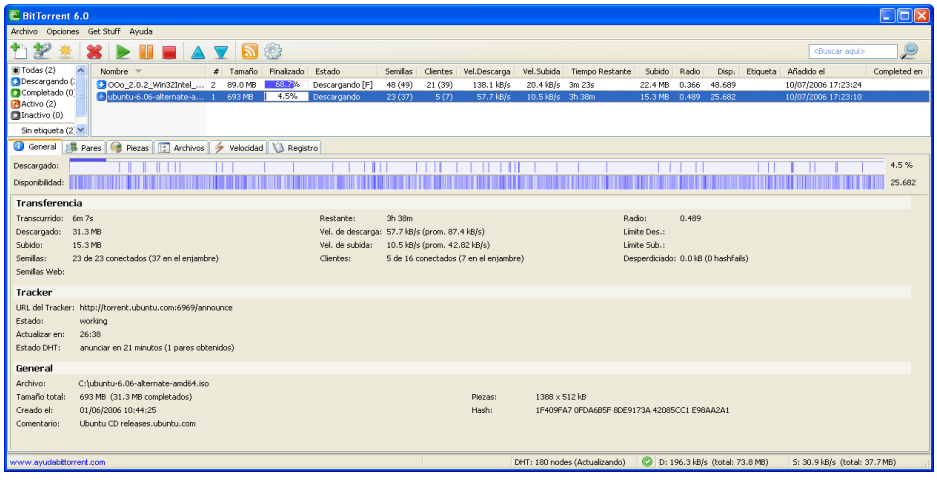


Figura 3.8. Interfaz del cliente BitTorrent Oficial

3.5. BITCOMET www.bitcomet.com

El cliente *BitComet* [22] fue denominado originalmente Cliente de SimpleBT. Está diseñado para Windows y programado en lenguaje C++. No es de código abierto.

BitComet puede descargar ficheros desde páginas web o servidores FTP, por lo que es un programa de descarga muy versátil.

Entre sus características destacan que soporta descargas simultáneas, redes de DHT (trackerless), habilidad para elegir prioridades en las descargas, opción de descargas seleccionadas en un paquete torrent, límites de velocidad y ajuste de ancho de banda, webseeding o previsualización del contenido durante la descarga.

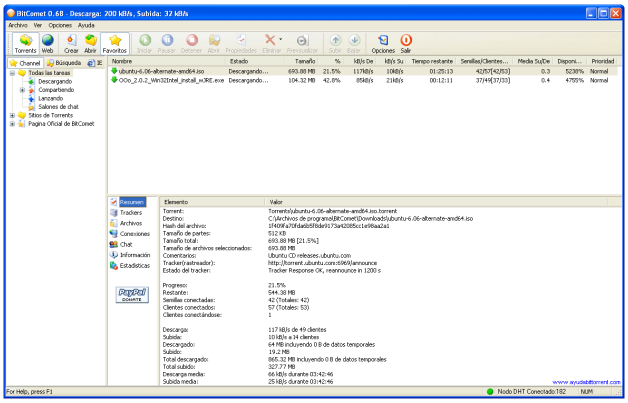


Figura 3.9. Interfaz del cliente BitComet

3.6. TRANSMISSION www.transmissionbt.com

Transmission [23] [24] es un cliente BitTorrent programado en C. Es de código abierto disponible bajo la licencia MIT, con algunas partes GPL. Se ejecuta, entre otros sistemas, en Linux y Mac pero no sobre Windows.

Este cliente es utilizado en menor medida que los otros descritos. Sus características son más limitadas aunque dispone de funciones como control de la velocidad de subida y bajada, limitación de velocidad en intervalos de tiempo, compatibilidad con UPnP, estadísticas y filtro para el intercambio con seeds seguros. Algunas de sus limitaciones son no disponer de un motor de búsqueda de torrents ni admitir webseeding ni RSS.

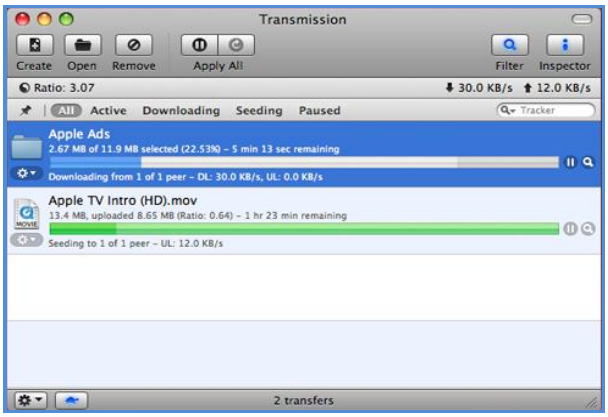


Figura 3.10. Interfaz del cliente Transmission

3.7. NUEVOS CLIENTES

Existe una nueva generación de clientes BitTorrent que no se encuentran entre los más utilizados en la actualidad debido a su reciente creación pero que ofrecen nuevas funcionalidades interesantes en el ámbito del Peer-to-Peer.

Entre ellos, se encuentra el cliente *Tribler* [25] desarrollado en 2008 por un equipo con el mismo nombre de la Universidad Técnica de Delft y la Universidad Vrije de Amsterdam. Su característica más importante es que permite realizar búsquedas descentralizadas sin depender de los trackers. Los usuarios pueden encontrar ficheros torrent existentes en otros usuarios de la red. Ofrece streaming P2P de las descargas que se están realizando. Incluye otras funciones como la ventaja de poder hacer rankings de los torrents y también de los usuarios. Los mejores torrents aparecerán primero en las búsquedas y los usuarios que más compartan tendrán más importancia.

Este mismo año se ha presentado otro cliente llamado *Chrysalis* [26] desarrollado por BitTorrent Inc. (al igual que los clientes *µTorrent* y *BitTorrent*) y disponible, por ahora, en una versión alfa sólo para Windows. El objetivo de este cliente es facilitar al usuario la tarea de descargar contenidos de la red. Surge de la idea de la compañía de considerar que *µTorrent* puede resultar complejo de usar para usuarios inexpertos. Ofrece una interfaz sencilla e intuitiva que sugiere diferentes descargas en su página principal ofreciendo contenidos cuando el usuario abre el programa.

Por otro lado, este año se ha presentado el cliente *BitMate* [27] perteneciente al Proyecto *Dritte* cuyo objetivo es impulsar el desarrollo mediante las tecnologías. Funciona a través de conexiones de acceso telefónico, de manera que se puede compartir y descargar torrents con un acceso de banda limitado. El cliente está implementado sobre *Vuze* al que realiza ligeras modificaciones en la forma en la que la aplicación interactúa con la red.

Estos clientes no son tan populares como los descritos en los anteriores apartados pero van captando usuarios al implementar características de gran interés para algunos grupos.

4. APLICACIÓN DESARROLLADA

Este capítulo desarrolla los detalles de la aplicación realizada en este Proyecto tanto de manera teórica, con la explicación de los sistemas implementados, como de manera práctica con un ejemplo de funcionamiento. Se pretende con ello generar una documentación que permita conocer todos los aspectos de la aplicación para la futura integración con otros módulos como el sistema de recomendación.

A continuación se definen las características del plugin, de los módulos que participan en el sistema completo y de los dos sistemas planteados en el Proyecto:

- Sistema de recomendación de contenidos
- Sistema de alerta de contenidos falsos

Por último, se desarrolla un ejemplo de funcionamiento y se presenta la organización y estructura de las carpetas y ficheros de los que consta la aplicación.

4.1. DESCRIPCIÓN

Vuze es el cliente BitTorrent para el que se ha implementado la aplicación de este Proyecto. Esto es posible debido a que este programa proporciona una interfaz de programación de aplicaciones (*API*) que permite crear complementos con los que interactuar.

La aplicación amplía la funcionalidad de *Vuze* recogiendo la información de las descargas que tengan en su librería los usuarios que instalen el plugin desarrollado. A partir de la información almacenada en la base de datos de la aplicación se pretenderá generar un sistema de recomendación de contenidos. Por otro lado, se proporciona un sistema de alerta de descargas con contenido falso a partir de los resultados de la aplicación *FakeTorrent* con la que se comunica.

La aplicación sigue un modelo cliente-servidor. El primero es el responsable de la instalación del plugin en *Vuze* desde el cual se recolectan los datos. El segundo es el receptor de esta información y el encargado de manejar la base de datos.

► CLIENTE

Los clientes son los usuarios de *Vuze* que dispongan del plugin y procedan a su instalación. Todos los parámetros de las descargas de su librería se enviarán entonces a un servidor que se encargará de mantener registrada dicha información. El número de clientes activos variará en el tiempo dependiendo del estado de sus sesiones.

Para poder tener identificados a los clientes que hayan tenido o tengan instalado el plugin se les asigna, durante la instalación, un identificador aleatorio de veinte bits a cada usuario. Esta *id* es guardada en un fichero de texto llamado *'id_usuario.txt'* que queda almacenado en el directorio por defecto del usuario.

El motivo de asociar a cada usuario un identificador es poder mantenerlos registrados ante cualquier acción para evitar la redundancia en la introducción de información en la base de datos, conseguir reducir la carga en la conexión a la base de

datos y almacenar de manera más eficiente la información en la misma. Este identificador se requiere también por la aplicación que detecta contenidos falsos.

Cuando un usuario desinstale el plugin y posteriormente lleve a cabo una nueva instalación se procede a comprobar si existe el fichero de texto. Si existe significa que el usuario ha tenido instalado el plugin con anterioridad, por lo que la base de datos a la que accede al servidor ya dispone de información en cuanto a sus sesiones y descargas y sólo se actualiza (*ver Figura 4.1*).

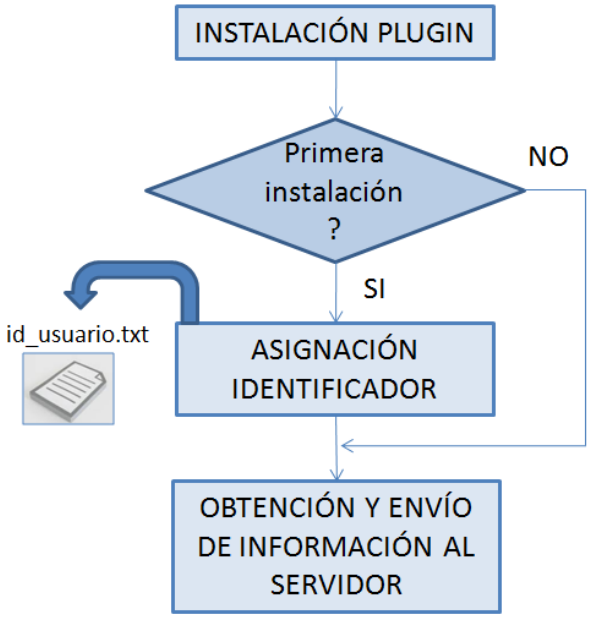


Figura 4.1. Diagrama de estados del arranque del plugin

► SERVIDOR

Para desarrollar la funcionalidad de la aplicación es indispensable contar con un servidor central encargado de recibir la información de todos los clientes que se comunican con él y de cumplir las necesidades requeridas para almacenar dicha información en una base de datos. Para ello se cuenta con una aplicación Java que se ejecuta desde un ordenador que, a su vez, puede actuar de usuario evitando la necesidad de contar con un servidor de dedicación exclusiva.

El servidor es la entidad que sirve de nexo entre los clientes que instalan el plugin y la base de datos que recoge la información de los mismos. De este modo, el sistema de recomendación de contenidos deberá integrarse con el servidor para realizar las recomendaciones a los usuarios a partir de la información de la que dispone.

Los detalles de cómo llevar a cabo el arranque del servidor se encuentran en el *ANEXO 2. Manual de puesta en marcha del servidor*.

Tras llevar a cabo el arranque del servidor éste necesita cuatro parámetros para su configuración y la de la conexión a la base de datos. Dichos parámetros se obtienen de un fichero de configuración con pares clave/valor llamado '*config_servidor.properties'*.

Al evitar que el valor de estos parámetros se especifique en los archivos *.java* del servidor se consigue disponer de ellos de manera agrupada en un fichero de propiedades que facilita el acceso en las futuras versiones que tengan que consultarlos o modificarlos.

Las claves existentes en el fichero son:

- → *Parámetro de configuración del servidor*
  - `servidor.puerto_a`: puerto TCP atendido por el socket servidor responsable de la recepción de información del cliente y sus descargas.

- → *Parámetros de conexión a la base de datos*
  - `bbdd.nombre`: Nombre de la base de datos que almacena los datos recogidos.
  - `bbdd.usuario`: Usuario.
  - `bbdd.contrasena`: Contraseña.

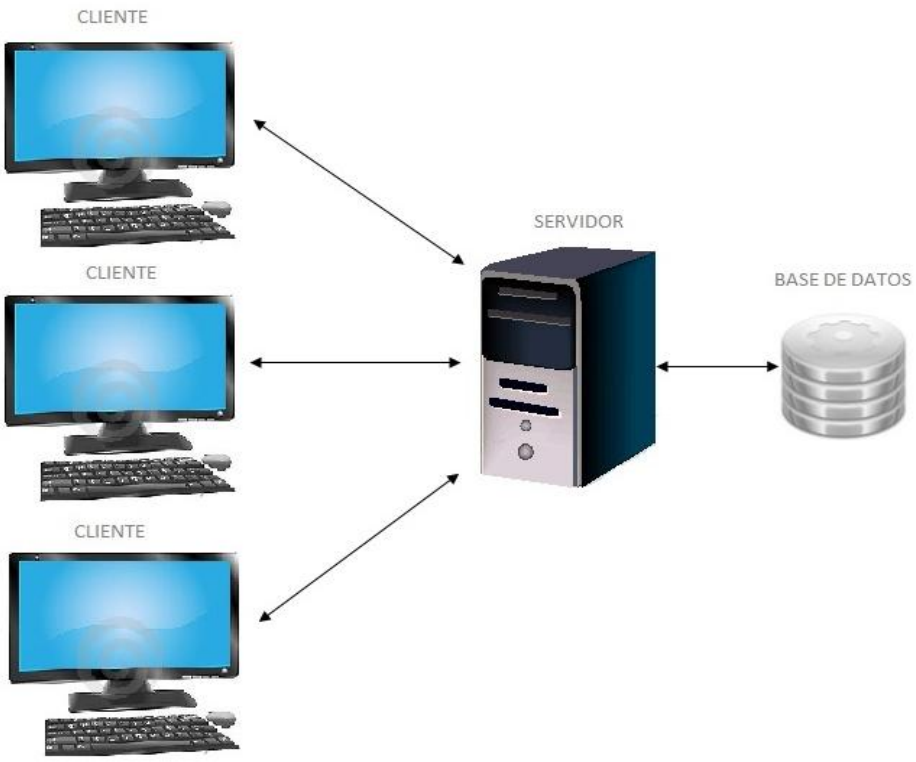


Figura 4.2. Escenario de la aplicación

4.2. PLUGIN

En este Proyecto se ha desarrollado el complemento *plugin_vuze_0.1.jar*, que se trata de un plugin instalable desde Vuze. A continuación se describen diversos aspectos:

❖ Instalación

Cada cliente interesado en la utilización del plugin debe acceder a la opción 'Instalar Complementos' disponible en la opción 'Herramientas' de la barra de menú de *Vuze* y marcar el método de instalación 'Mediante un archivo'. Aparece un desplegable donde se selecciona el archivo del plugin llamado: *plugin_vuze_0.1.jar*. Para una descripción detallada de cómo realizar la instalación consultar el *ANEXO 3. Manual del plugin para el usuario*.

A partir de que la instalación del plugin se complete, el cliente reporta al servidor la información de sus descargas actuales y las novedades que haya a partir de ese momento.

❖ Página de configuración

El plugin cuenta con una página de configuración, accesible siguiendo los pasos detallados en el *ANEXO 3. Manual del plugin para el usuario*, que permite la modificación de ciertos valores en el supuesto de querer variar los existentes por defecto. En la página de configuración (*ver Figura 4.3*) se incluye también como ayuda al usuario un enlace a la guía para usuarios de *Vuze* disponible en la sección de Ayuda de su página oficial.

Los parámetros que pueden ser modificados son los que permiten la comunicación con el servidor:

- *Dirección IP del servidor*

  Esta variable podría requerir ser modificada ante el cambio de la dirección IP que atiende el servidor. Es imprescindible este parámetro para poder existir la comunicación entre cliente y servidor.

- *Puerto*

  El servidor atiende las peticiones en el puerto indicado en este parámetro por lo que su modificación podría ser necesaria en determinadas situaciones.

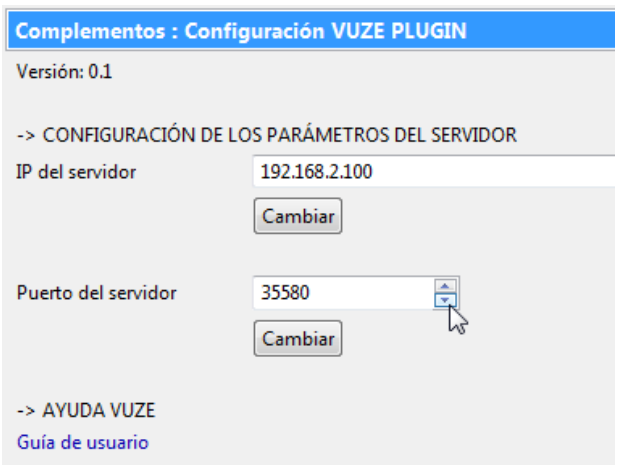


Figura 4.3. Apariencia de la página de configuración del plugin

Vuze ofrece la posibilidad de anunciar al usuario mediante mensajes (de alerta, información o error) detalles que ocurren durante la utilización del programa. Esta herramienta es aprovechada por el plugin para hacer comunicaciones al usuario de información como la tratada en la página de configuración.

Cuando el usuario lleva a cabo modificaciones se le alerta de estos cambios mostrándole un mensaje informativo indicando las variaciones que ha realizado en los parámetros correspondientes (*ver Figura 4.4*).

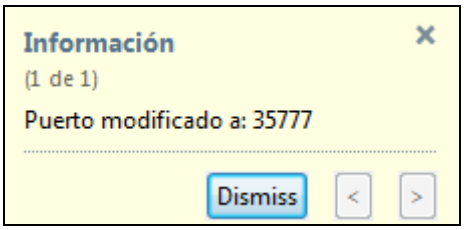


Figura 4.4. Ejemplo de una alerta informativa de variación de parámetros

❖ Soporte multi-lenguaje

Vuze hace uso de los paquetes de recursos, lo que significa que usa el mismo mecanismo para soportar otros idiomas lo que permite implementar un sistema multi-lenguaje [28]. Para ello, el plugin desarrollado cuenta con ficheros de propiedades que definen los textos del plugin en varios idiomas.

Con esta funcionalidad, se tienen los recursos localizados e identificados facilitando las traducciones en función de la configuración local de idioma de cada cliente.

Para acceder desde *Vuze* a la sección donde se permite modificar el idioma, hay que entrar en las *Herramientas/Opciones/Interfaz/Idioma* (*ver Figura 4.5*).

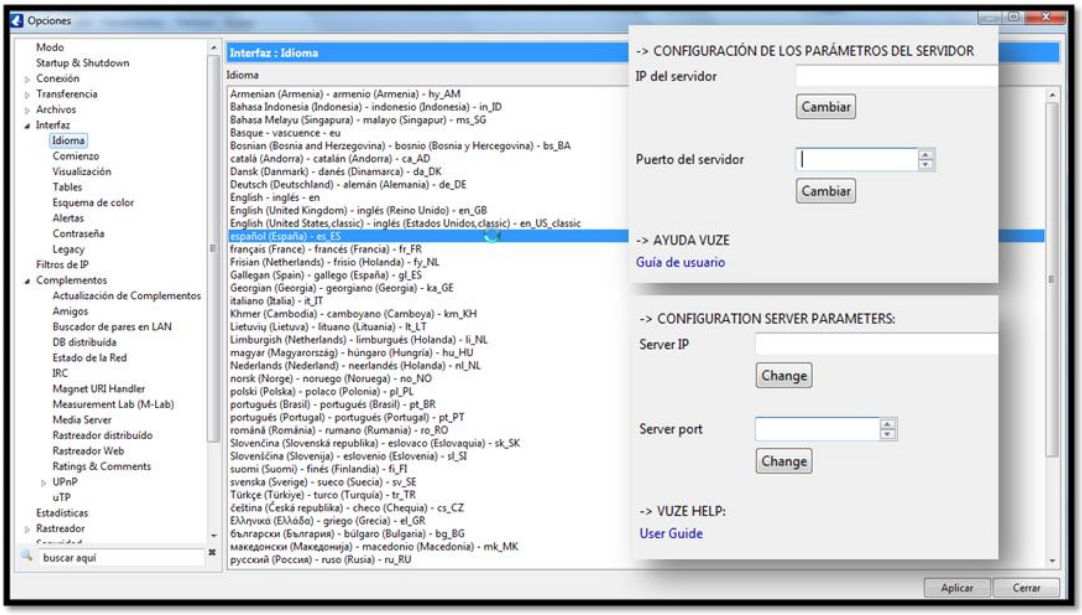


Figura 4.5. Selección del idioma del plugin y página de configuración en español e inglés

4.3. SISTEMA DE RECOMENDACIÓN DE CONTENIDOS

La tarea fundamental de la aplicación desarrollada es la recolección de la información sobre las descargas de los clientes de Vuze que servirá para desarrollar un sistema de recomendación de contenidos basado en coincidencias entre usuarios.

La entrada a este sistema (*ver Figura 4.6*) estará formada por la información característica del usuario activo, pero también de la información del resto de usuarios del sistema, que actúan como colaboradores. En este sentido, la realimentación por parte de los usuarios es muy importante de cara a albergar una información más completa ante futuros procesos de generación de recomendaciones. La salida del sistema estará basada en el resultado de un método de generación de recomendaciones.

Figura 4.6. Esquema del proceso de generación de una recomendación

En apartados posteriores se detalla la información que se extrae del cliente que instala el plugin, pero es necesario conocer qué parámetros se requieren para comprender el diseño de la base de datos:

- De cada usuario:
  - Identificador de usuario
  - Timestamp de cada instante de conexión
  - Timestamp de cada instante de desconexión
- De cada descarga:
  - Identificador de torrent
  - Nombre del torrent
  - Nombre del fichero
- De cada relación usuario-descarga:
  - Identificador de usuario
  - Identificador de torrent
  - Estado de la descarga
  - Timestamp del inicio de la descarga
  - Timestamp del fin de la descarga

A continuación se describen diferentes aspectos que permiten recoger esta información de manera eficiente y logran mantenerla actualizada.

4.3.1. BASE DE DATOS

► *Diseño*

El diseño de la base de datos tiene como objetivo principal generar las tablas que modelarán los registros donde se guarde la información a partir de la cual se desarrollará el sistema de recomendación. Se pretende dotar a la base de datos de flexibilidad para que ésta crezca sin la necesidad de ser reestructurada.

Para obtener una recuperación rápida y eficiente de la información se requiere que los datos almacenados lo hagan sin redundancia. Por lo tanto, para conseguir un procesamiento eficaz de la información se persiguen los siguientes objetivos:

- Controlar la redundancia de la información.
- Evitar pérdidas de información.
- Dotar de capacidad para representar toda la información.
- Mantener la consistencia de los datos.

El modelo al que se ha recurrido para la base de datos del Proyecto es el relacional ya que se ajusta a las necesidades requeridas en el almacenamiento de la información empleando conjuntos de datos interconectados entre sí. Dada la necesidad de que la información almacenada sea editada en el tiempo mediante operaciones como actualización, inserción o borrado, se requiere que la base de datos sea dinámica.

Se ha recurrido a una herramienta de modelado de datos como es el *Diagrama Entidad-Relación* para conocer qué entidades son relevantes así como sus relaciones y propiedades (*ver Figura 4.7*).

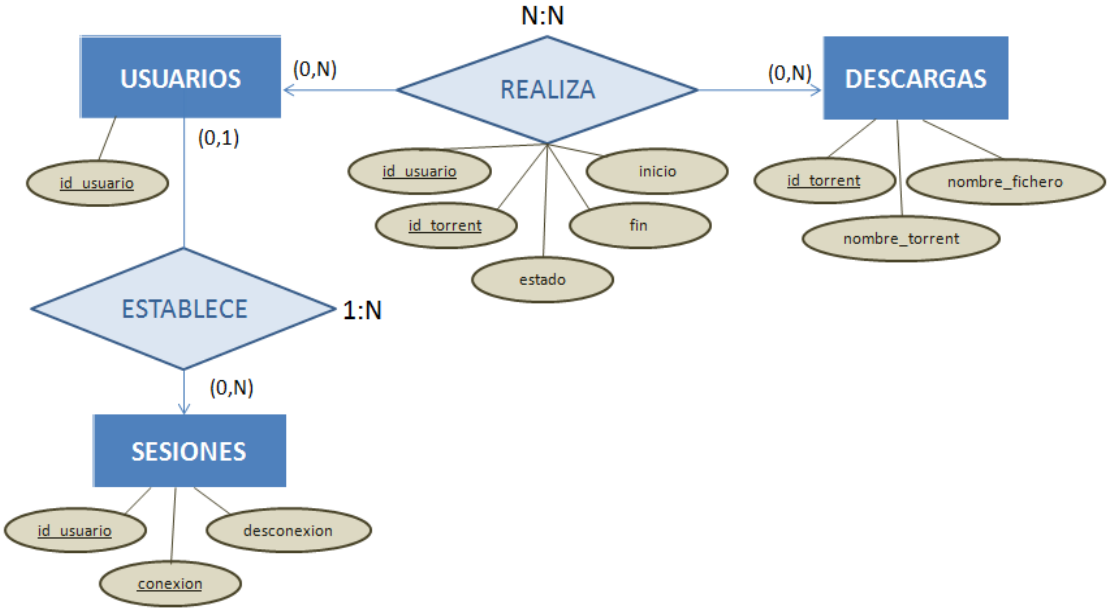


Figura 4.7. Diagrama Entidad-Relación de la base de datos

Según los requisitos establecidos la base de datos relacional está compuesta de cuatro tablas cuyos elementos y relaciones figuran en el grafo de la *Figura 4.8* donde se muestran en negrita las claves primarias y con flechas las relaciones entre tablas.

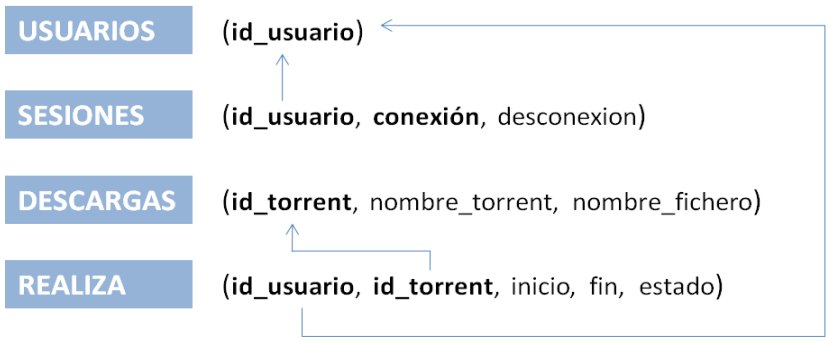


Figura 4.8. Grafo relacional de la base de datos

Las bases de datos relacionales almacenan la información en tablas con filas y columnas. La empleada en este Proyecto consta de cuatro tablas: **usuarios**, **sesiones**, **descargas** y **realiza**.

- **usuarios**: Tabla principal de la base de datos que cuenta con el campo *id_usuario* donde se almacenarán los identificadores de los clientes.

Figura 4.9. Atributo de la tabla usuarios

- **sesiones:** Tabla formada por tres registros que son: *id_usuario*, *conexion* y *desconexion*. Esta tabla guarda un historial de sesiones de cada cliente donde figura por cada sesión el identificador de usuario junto al timestamp de conexión y desconexión.

Figura 4.10. Atributos de la tabla sesiones

- **descargas**: Cada elemento de esta tabla representa una descarga con los parámetros que la caracterizan. Cuenta con los siguientes campos:
    - *id_torrent*: identificador del torrent (*infohash*).
    - *nombre_fichero:* nombre, incluida la extensión, del fichero a descargar.
    - *nombre_torrent:* nombre, incluida la extensión *.torrent*, del torrent.

Figura 4.11. Atributos de la tabla descargas

- **realiza:** Tabla asociada a la relación entre las entidades *usuarios* y *descargas* que guarda la información de las descargas dependiente del usuario y el identificador del torrent correspondiente:
    - *id_usuario*: identificador del cliente al que corresponde la descarga.

- *id_torrent*: identificador del torrent (*infohash*).
- *inicio*: timestamp del comienzo de descarga.
- *fin*: timestamp del fin de descarga.
- *estado*: registro que contendrá el entero (0 descargando, 1 descargado ó 2 cancelado) que indica la situación actual de la descarga en cuestión.

Figura 4.12. Atributos de la tabla realiza

► *Relaciones entre tablas*

Las claves que establecen las relaciones entre los campos de las tablas de la base de datos del Proyecto son las siguientes:

- *Claves primarias*:

La clave primaria de una tabla es una columna o conjunto de columnas que se escogen para identificar sus tuplas de modo único. Por lo cual, estas claves identifican unívocamente a cada fila de la tabla.

En la tabla *usuarios* la clave primaria es el campo *id_usuario* (identificador de cada usuario). Este identificador es único y característico de cada usuario. En la tabla *sesiones* la clave primaria está compuerta por el campo *id_usuario* y *conexión*. La tabla *descarga* cuenta con el atributo *id_torrent* como clave primaria. Las claves primarias de la tabla *realiza* son los identificadores de usuario y de torrent (*id_usuario* e *id_torrent*) ya que la combinación de estos campos identifica de manera única a cualquier fila de la tabla siendo imposible así que un registro esté duplicado.

- *Claves foráneas:*

La clave foránea o ajena está formada por una o varias columnas que están asociadas a una clave primaria de otra o de la misma tabla. De este modo, las referencias se crean para vincular o relacionar la información.

En este Proyecto ha sido conveniente relacionar las conexiones de un usuario (tabla sesiones) como las descargas que tiene asociadas (tabla descargas) al usuario al que correspondan. Por lo que la tabla *realiza* cuenta con dos claves ajenas (*id_usuario* e *id_torrent*) y la tabla *sesiones* con la clave ajena *id_usuario*.

► *Implementación*

Las tablas que componen la base de datos se crean empleando el lenguaje estándar de programación SQL. Se crean desde la consola que proporciona el gestor de bases de datos MySQL como interfaz entre la base de datos, el creador de la misma y el servidor (aplicación que la utiliza). Desde ahí se especifican también las relaciones entre los datos. Para su gestión desde el lenguaje Java se utiliza la API JDBC.

Para simplificar la creación de las tablas desde la máquina que vaya a ejecutar el servidor, se proporciona junto a la documentación del Proyecto el archivo *datos_plugin.sql*. De esta manera desde el gestor de base de datos se pueden importar las tablas. En el apartado 'Importación base de datos' del *ANEXO 2. Manual de puesta en marcha del servidor* se detallan las operaciones que hay que llevar a cabo para realizar la importación.

4.3.2. TRATAMIENTO DE INFORMACIÓN

► *Obtención de parámetros*

Desde que la instalación del plugin se completa en un cliente, el servidor registra todos los cambios en sus descargas. El plugin se encarga de este hecho recogiendo diversos parámetros de las descargas que tenga registradas el usuario en su librería de *Vuze*.

Para cada descarga el plugin obtiene los siguientes parámetros:

- *Identificador torrent* (*infohash*): Array de veinte bytes que identifican unívocamente a cada fichero torrent. Para facilitar su manejo se han transformado a una cadena con los correspondientes cuarenta caracteres ASCII.
    - Ejemplo: *906e3f8c10e84e465763eefe79e9472e6aa90020*
- *Nombre fichero*: Nombre y extensión del fichero correspondiente a la descarga.
    - Ejemplo: *eclipse-SDK-3.6-win32-x86_64.zip*
- *Nombre archivo .torrent*: Nombre y extensión *.torrent* de dicho archivo.
    - Ejemplo: *eclipse-SDK-3.6-win32-x86_64.zip.torrent*
- *Timestamp inicio*: Fecha y hora del instante en el que se ha iniciado la descarga.
    - Ejemplo: *Fri Feb 18 18:01:53 CET 2011*
- *Timestamp fin*: Fecha y hora del instante en el que ha finalizado la descarga.
    - Ejemplo: *Fri Feb 22 19:52:13 CET 2011*
- *Estado*: Número ('0','1' ó '2') que refleja la situación actual de cada descarga pudiendo variar entre los tres siguiente estados:
    - **0**: descargando. La descarga está en proceso.
    - **1**: descargado. La descarga ha sido finalizada con éxito.
    - **2**: cancelado. La descarga ha sido interrumpida antes de finalizar.

Los campos que reflejan el timestamp de inicio y/o fin de una descarga pueden tomar el valor de 'INICIO_DESCONOCIDO' y 'FIN_DESCONOCIDO' respectivamente. Esto ocurre si la descarga a la que van asociados comienza y/o se completa estando el plugin desinstalado ya que en esa situación se desconocen los valores. De esta manera, se evita almacenar información que no se ajuste a la realidad como sería guardar el instante de inicio o fin del momento en el que se descubriera la descarga para el plugin ya que no correspondería con el instante real.

► *Envío de parámetros*

El cliente envía los parámetros, correspondientes a la información de los usuarios y sus descargas, al servidor para su posterior gestión en el mismo.

La base de datos que gestiona el servidor con la información recogida por los clientes tiene que actualizarse en tiempo real para llevar un control preciso de los mismos. Para cumplir con esta necesidad, se necesita un mantenimiento continuo del estado de dicha información teniendo actualizados los datos.

Por lo tanto, es necesario que el cliente informe al servidor cuando instale y desinstale el plugin, de las nuevas descargas añadidas y de las novedades en las ya existentes. Los casos en los que los clientes establecen una comunicación con el servidor (*ver Figura 4.13*) para notificarle una nueva situación en su estado son los siguientes:

- ***Plugin instalado***: Cada instalación de un cliente se notifica al servidor distinguiendo dos situaciones:

  1. Primera instalación

  La primera vez que un usuario instala el plugin en su programa *Vuze* se lo notifica al servidor enviándole el identificador de usuario que se le ha asignado y todos los parámetros de las descargas que están actualmente en su lista de *Vuze*.

  2. Reinstalación

  Si en la instalación del plugin se detecta que existe en la máquina del usuario el fichero de texto con un identificador que se genera en la primera instalación significará que el cliente ya lo ha tenido instalado con anterioridad. En esta ocasión, junto al identificador de usuario y los parámetros de las descargas actuales en *Vuze*, se envía al servidor un aviso de la reinstalación para que lo tenga en cuenta a la hora de gestionar la base de datos.

- ***Plugin desinstalado*:** Si se produce una desinstalación del plugin en un cliente, éste envía al servidor un aviso indicando que deja de tener instalado el plugin, su identificador de usuario y el timestamp del instante en el que ha ocurrido.

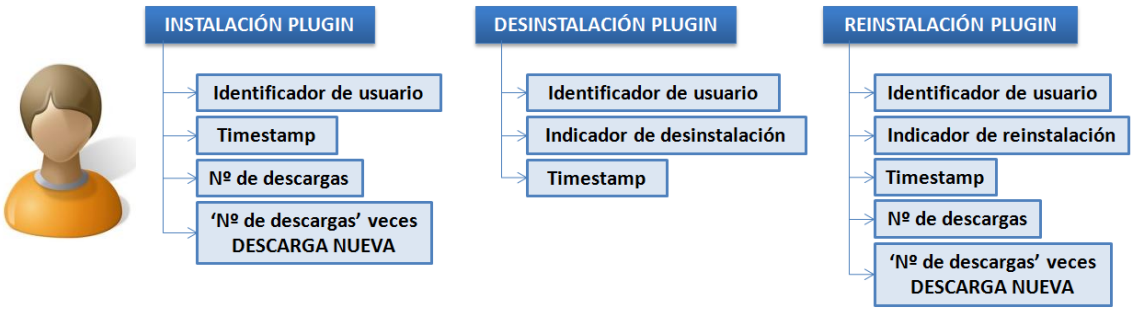


Figura 4.13. Parámetros enviados por el cliente en las operaciones de instalación.

Las posibles situaciones que pueden darse en las descargas de los usuarios que conllevan una notificación al servidor por parte del cliente son:

- ***Nueva descarga:*** Si se incorpora una nueva descarga a la lista del cliente, el plugin se encarga de obtener los parámetros correspondientes a la misma y de informar de ellos al servidor. Junto a esos parámetros le envía el identificador del usuario al que corresponde la nueva descarga y un indicador avisando de la nueva incorporación.

- ***Actualización del estado en una descarga existente:*** Cuando el estado de una descarga varía a completo o cancelado se le notifica el cambio al servidor para que actualice los campos correspondientes en la base de datos. Los dos casos posibles son:

  1. Descarga eliminada

  Se envía al servidor el identificador del usuario al que corresponde la descarga eliminada, el identificador de torrent (*infohash*) de la descarga, el nuevo estado de la misma y un indicador que avise al servidor del cambio.

  2. Descarga completa

  Se envía al servidor los mismos parámetros que en el caso *1. Descarga eliminada* y además el timestamp del instante en el que ha concluido la descarga.

Según lo descrito, el envío de la información de las descargas está compuesto por diferentes parámetros en función de la situación concreta según indica la siguiente figura:

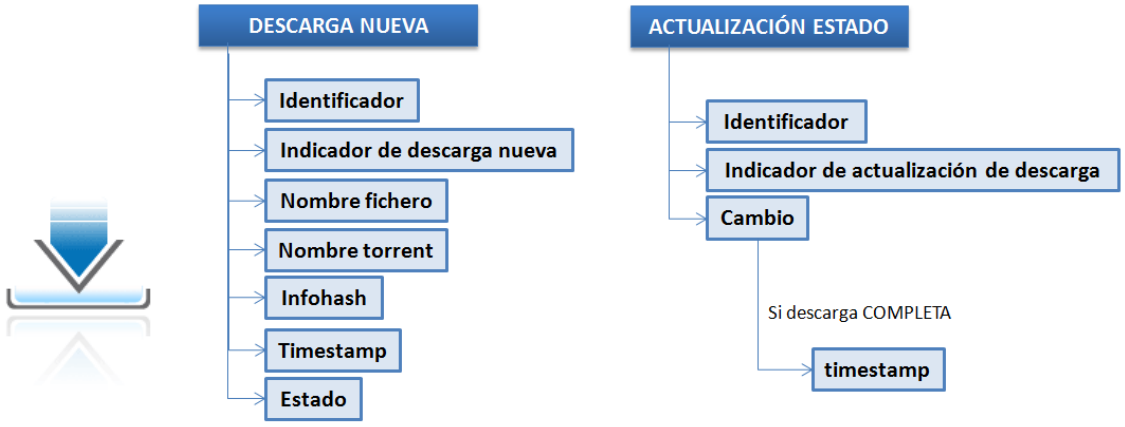


Figura 4.14. Parámetros enviados por el cliente en las operaciones con descargas

► *Tratamiento de parámetros*

Ya en la parte del servidor, se realiza el mantenimiento de la información de los usuarios para disponer de una base de datos actualizada. Las operaciones llevadas a cabo sobre la base de datos desde el servidor son de tres tipos:

→ *Inserción*

Cuando el servidor recibe los parámetros correspondientes a una nueva descarga procede a añadirlos en la base de datos.

→ *Modificación*

Tras un aviso recibido en el servidor notificando que ha habido un cambio en el estado de alguna descarga ya existente.

→ *Consulta*

El servidor requiere en ciertas ocasiones conocer las descargas registradas en la base de datos para un determinado usuario o si existe una descarga identificada por su *infohash* para un usuario concreto.

Para realizar las anteriores operaciones de manera acorde a cada situación (*ver Figura 4.15*), cada vez que el servidor recibe datos de un cliente comprueba si éste ha sido registrado en la base de datos con anterioridad. Para comprobarlo se consulta si la *id* del usuario que va a realizar un envío de datos se encuentra ya en la base de datos.

Si la comprobación de existencia de usuario es negativa se crea el registro del nuevo usuario en la base de datos almacenando su listado completo de descargas. En cambio, si el cliente ya existe, se procede a realizar con los datos la operación que corresponda entre las siguientes:

- **Inserción de una nueva descarga**:

  Se distinguen dos casos dependiendo de si es una descarga desconocida para la base de datos, o si en cambio, ya figura para otro usuario. Si es así, no se guardan en la base de datos los datos que caracterizan a la descarga sino sólo los parámetros que la asocian a un usuario (estado, timestamp de inicio y timestamp de fin).

- **Modificación del estado de una descarga:**

  Si el cambio producido en una descarga es debido a que se ha completado se almacena el timestamp de ese instante.

- **Modificación de la sesión de un usuario por desinstalación del plugin:**

  Se guarda el timestamp del instante en el que se produce la desinstalación para el usuario.

- **Reinstalación del plugin:**

  Se actualiza la sesión del usuario y se lleva a cabo una actualización de los parámetros de sus descargas según el siguiente orden:

  1. Lectura de las descargas que se encontraban ya registradas para ese usuario actualizando el estado de las mismas en el caso de que haya variado en el tiempo transcurrido (si se ha completado o cancelado). Así mismo, se registra el timestamp de finalización en el caso de que alguna descarga haya concluido.

  2. Inserción de las descargas que han sido añadidas a la librería durante el tiempo que el plugin ha estado desinstalado.

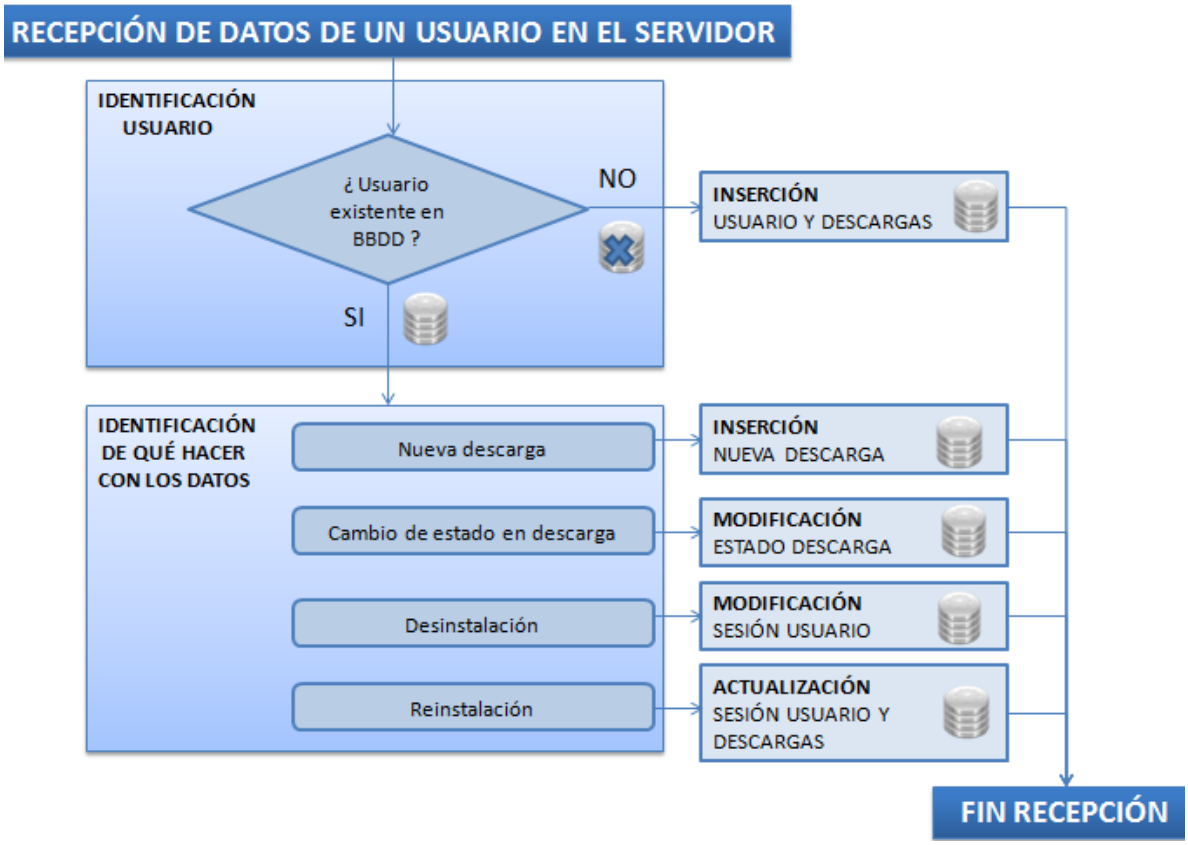


Figura 4.15. Diagrama con la dinámica de tratamiento de datos en el servidor

4.3.3. COMUNICACIÓN ENTRE MÓDULOS

► *Comunicación Cliente - Servidor*

La comunicación entre los clientes y el servidor es imprescindible para conseguir el objetivo de recoger los datos de los clientes. Por ello, se ha empleado una comunicación mediante sockets [29] [30] como solución para comunicar procesos, que se encuentran en distintas máquinas, a través de la red. Con los sockets se consigue un sistema de diálogo entre procesos que permite que éstos intercambien información entre sí.

- Procedimiento

El servidor se ejecuta en una máquina específica y tiene un socket que responde a un puerto concreto. El servidor únicamente espera a que un cliente quiera enviar información escuchando a través del socket quedándose bloqueado hasta que algún cliente se conecte.

Por su parte, el cliente conoce la dirección IP de la máquina en la cual el servidor se está ejecutando y el número de puerto al que está conectado el servidor. Para iniciar una comunicación con el servidor, el cliente intenta encontrar al servidor en la máquina y puerto especificado. Si no surge ningún problema, el servidor acepta la conexión y atiende la petición del cliente. Hasta que no recibe todos sus datos no vuelve a escuchar peticiones de otros clientes, es decir, no acepta simultáneamente varios clientes.

En el lado del cliente, si la conexión es aceptada, se crea un socket que se usa para comunicarse con el servidor. Ahora el cliente y el servidor pueden comunicarse escribiendo o leyendo los flujos de datos desde sus respectivos sockets.

Al finalizar la comunicación el cliente cierra el socket abierto y el servidor vuelve a quedarse a la espera para atender nuevas peticiones. La siguiente figura representa todo el proceso descrito:

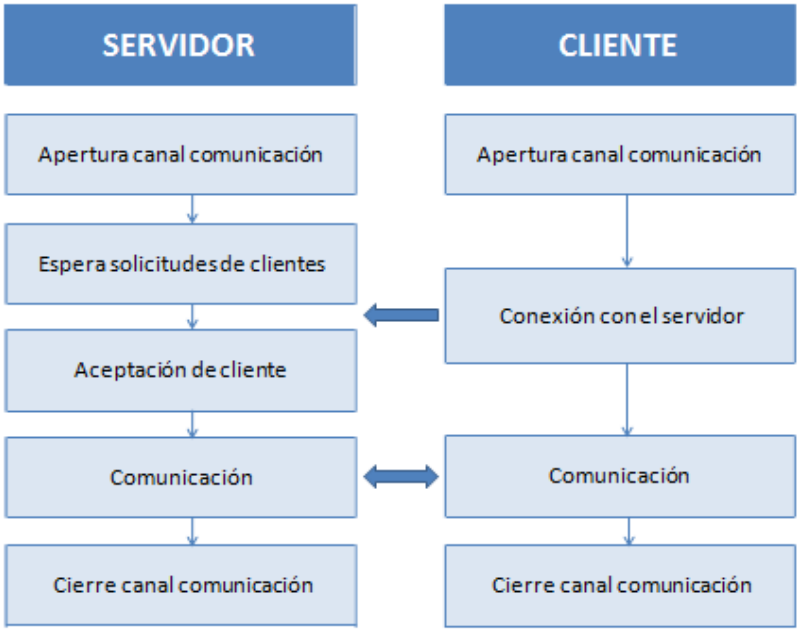


Figura 4.16. Mecánica de la comunicación entre servidor y clientes

- Parámetros característicos del servidor

El puerto que atiende un servidor y al que debe conectarse el cliente para comunicarse puede ser cualquier entero entre 1 y 65535. Los números del 1 al 1023 están reservados para servicios del sistema como ftp, mail, telnet, etc. Por lo que se ha escogido el 35555 como puerto que atiende el servidor al que se conectarán todos los clientes que le envíen datos: `servidor.puerto_a` → 35555.

Este valor viene establecido por defecto pero, si fuera necesario, es modificable desde la página de configuración del plugin según se detalla en el apartado 4.2. PLUGIN. Del mismo modo ocurre con la dirección IP de la máquina donde se ejecuta el servidor a la que realizan las peticiones los usuarios.

- Implementación en Java

Cuando se escriben programas Java que se comunican a través de la red, se está programando en la capa aplicación. Java proporciona el paquete *java.net* que evita la necesidad de trabajar con las capas TCP y UDP. Estas clases proporcionan comunicación de red independiente del sistema.

A partir de las clases del paquete *java.net*, los programas Java pueden utilizar TCP o UDP para comunicarse a través de Internet. Las clases *URL*, *URLConnection*, *Socket*,

y *SocketServer* utilizan TCP mientras que las clases *DatagramPacket* y *DatagramServer* utilizan UDP.

TCP proporciona un canal de comunicación fiable punto a punto, lo que se emplea en las comunicaciones de las aplicaciones cliente-servidor como la implementada en este Proyecto.

Por otro lado, para gestionar las operaciones entrada/salida a través de streams, Java proporciona el paquete *java.io*. Por lo tanto, la aplicación desarrollada emplea los paquetes: *java.io* y *java.net*:

En el cliente:

- *InetAddress*

  Clase utilizada para manipular las direcciones IP.

- *Socket*

  Esta clase proporciona métodos para la entrada/salida a través de streams que permiten la lectura y escritura. Es el objeto básico en toda comunicación a través de Internet bajo el protocolo TCP.

- *DataOutputStream*

  Con este stream de salida el cliente envía la información al socket del servidor.

En el servidor:

- *ServerSocket*

  Objeto utilizado en las aplicaciones servidor para escuchar las peticiones que realicen los clientes a los que está conectado. A través de él se lleva a cabo toda la comunicación.

- *DataInputStream*

  Esta clase se emplea para recibir las entradas de datos que se produzcan de los clientes que se hayan conectado al servidor.

► *Comunicación Servidor - Base de datos*

La gestión de los datos recogidos por el servidor requiere una comunicación entre dicho servidor y la base de datos. Se ha necesitado una herramienta que permitiera almacenar, modificar y consultar parámetros en la base de datos desde el programa servidor. La librería estándar JDBC ha permitido realizar estas operaciones. Para ello, se ha recurrido a la biblioteca de conexión apropiada al modelo de la base de datos, y se ha accedido a ella estableciendo una conexión a través del localizador de la base de datos y los parámetros de conexión específicos.

De esta manera, se dispone del paquete `java.sql`, en el que existen las clases que permiten trabajar con las bases de datos. Las clases que se han empleado en este Proyecto con su función correspondiente son:

- *DriverManager* Carga del driver JDBC
- *Connection* Establecimiento de las conexiones
- *PreparedStatement* Ejecución de sentencias SQL precompiladas
- *Statement* Ejecución de sentencias SQL
- *ResultSet* Almacenamiento del resultado de las consultas

Una vez conocido el paquete necesario para la comunicación entre el servidor y la base de datos se pasan a describir a continuación las indicaciones para realizar la conexión [31]:

1. Carga del driver JDBC: `com.mysql.jdbc.Driver`
2. Establecimiento de la conexión especificando los siguientes parámetros de la base de datos:
    a. URL que especifica el protocolo de JDBC, un subprotocolo y el dominio del sistema al que se accede: `jdbc:mysql://localhost/datos_plugin`
    b. Nombre del usuario habilitado para entrar en el sistema.
    c. Contraseña del usuario.
3. Ejecución de consultas SQL y procesamiento de resultados.
4. Liberación de los recursos empleados.
5. Manejo de las excepciones que se puedan producir en el proceso.

4.4. SISTEMA DE ALERTA DE CONTENIDOS FALSOS

Se ha desarrollado un sistema de aviso de contenidos falsos al usuario de *Vuze*. La implementación del mecanismo que determina la falsedad de un contenido publicado no ha sido objeto del Proyecto por lo que el sistema desarrollado se ha integrado con la aplicación *FakeTorrent* realizada en el *Departamento de Ingeniería Telemática*. De este modo, el sistema de alerta de contenido falso proporcionado por el plugin se basa en los resultados reales ofrecidos por dicha aplicación.

4.4.1. INTEGRACIÓN CON LA APLICACIÓN *FAKETORRENT*

La comunicación entre el sistema desarrollado en este Proyecto y la aplicación *FakeTorrent* se realiza cada vez que el plugin registra una nueva descarga, bien porque acaba de ser instalado y dispone ya de alguna en su librería o porque se añada cuando ya está instalado. Para ello, el cliente envía una petición solicitando la información y la aplicación le responde con el resultado de determinar si es fake o no. En base al resultado, el plugin muestra una alerta de carácter informativo (*ver Figura 4.16*).

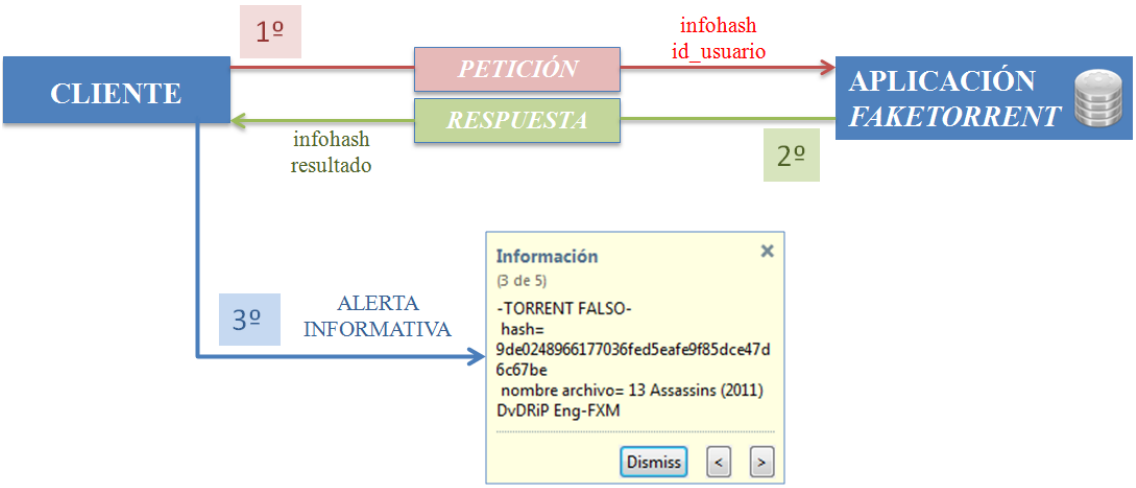


Figura 4.17. Esquema con la comunicación entre cliente y aplicación.

- *Petición*

La comunicación con la aplicación se realiza mediante una petición a la misma en la que se incluye el identificador de torrent a consultar y el identificador de usuario asociado al cliente que está realizando la petición.

Sin embargo, tras la instalación inicial del plugin la comprobación de contenidos falsos se realiza de manera masiva en una única petición. En ésta se envían todos los identificadores de los torrents que se encuentren en la librería de descargas.

- *Respuesta*

La aplicación consulta en la base de datos de la que dispone el resultado referido a cada hash de los que tiene analizados y devuelve un fichero xml con los resultados. Dicho fichero tiene la siguiente estructura:

```
<?xml version="1.0" encoding="UTF-8" ?>
<torrents>
   <torrent hash=”Hash URL enconded” result=res>
   …
   <torrent hash=”Hash URL enconded” result=res>
</torrents>
```

El servidor analiza la respuesta extrayendo de ella el resultado de las consultas sobre los distintos hash. Las posibles respuestas recibidas por parte de la aplicación son:

- **1**: Torrent falso
- **2**: Torrent verdadero
- **3**: Torrent conocido por la aplicación pero sin determinar si es falso
- **4**: Torrent no controlado

4.4.2. SISTEMA DE ALERTA

Uno de los objetivos del Proyecto ha sido informar al usuario de *Vuze* que instalara el plugin de posibles torrents falsos que se encuentren en su librería de descargas en el momento de la instalación del plugin, o bien, que añada como nueva descarga en un determinado instante. En el caso de que alguna de las descargas de las que dispone el cliente sea falsa, en proceso o ya finalizadas, se muestra una alerta en su cliente *Vuze*. La comprobación se realiza de igual manera cuando se añade una descarga mientras el plugin está instalado.

Para ello, el plugin cuenta con un sistema de aviso de contenidos falsos mediante alertas en Vuze permitiendo que el usuario conozca si ha descargado o va a descargar un fake. Dado que el resultado devuelto por la aplicación *FakeTorrent* puede ser de cuatro tipos, la alerta que se muestre al usuario será acorde a dicho resultado (*ver Figura 4.18*).

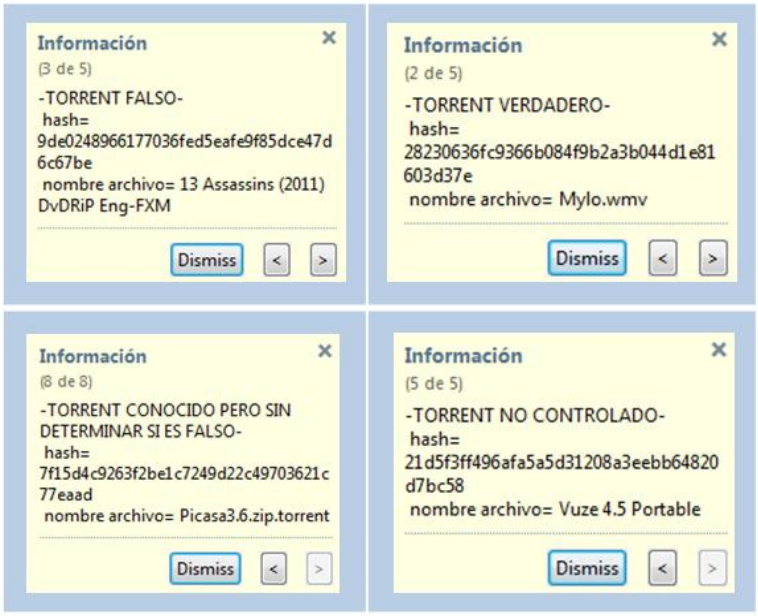


Figura 4.18. Ejemplos de alertas informativas sobre contenidos falsos

4.5. FUNCIONAMIENTO

En este capítulo se detalla el modo de funcionamiento de la aplicación atendiendo a las diferentes situaciones que podrían ocurrir durante la ejecución de la misma.

Para ello, en el primer apartado se especifican las operaciones que se llevan a cabo en el programa servidor, en los clientes del mismo y en la base de datos. El segundo apartado detalla un ejemplo de uso de la aplicación en el que ocurren todas las operaciones descritas en el primero con los resultados que desencadena cada una.

4.5.1. OPERACIONES

- *Arranque del servidor*

La aplicación debe comenzar arrancando el servidor en una máquina según indica el *Anexo 2. Manual de puesta en marcha del servidor*. La ejecución de este programa implica que el servidor:

- Carga el driver JDBC y establece la conexión con la base de datos estando preparado desde entonces para realizar cualquier operación sobre la misma.
- Establece un socket que queda a la espera de peticiones por parte de algún cliente.

- *Instalación del plugin en el cliente*

Desde *Vuze* cada cliente instala el plugin según las indicaciones del *Anexo 3. Manual del plugin para el usuario*. Cuando la instalación se completa el cliente ya está preparado para comunicarse con el servidor.

- *Comunicación cliente-servidor*

Tras la instalación del plugin el cliente envía al servidor los parámetros de las descargas que se encuentran en ese momento en su librería de *Vuze*.

- *Comunicación servidor-base de datos*

La información recibida en el servidor es almacenada en una base de datos a la cual accede el servidor actuando como encargado de actualizar y gestionar la misma.

- *Actualización datos del cliente*

Una vez instalado el plugin y enviada la información inicial, el servidor vuelve a quedar a la espera de nuevos datos que serán enviados por el mismo cliente (cuando haya nuevas descargas, se completen las existentes, se produzcan cancelaciones o el usuario decida desinstalar el plugin) o por otros clientes que instalen o tengan instalado el plugin.

Si se produce una actualización se vuelven a comunicar el cliente con el servidor y éste con la base de datos pero únicamente enviando la información de la descarga o usuario que ha sido actualizado.

- *Desinstalación del plugin*

El cliente puede desinstalar el plugin cuando lo desee desde *Vuze*. Este hecho implica una notificación al servidor que al enterarse actualiza el historial de sesiones para el cliente en cuestión.

- *Reinstalación del plugin*

Es posible que tras la desinstalación del plugin por parte de algún cliente, éste decida volver a instalarlo. En ese caso, el servidor es informado de que el nuevo cliente se trata realmente de uno que ya ha sido registrado anteriormente. Por ello, esta situación implica únicamente una actualización de los datos disponibles para dicho cliente en la base de datos y una incorporación de las novedades.

4.5.2. EJEMPLO

A continuación se presenta un ejemplo de funcionamiento de la aplicación atendiendo a las diversas situaciones que pudieran ocurrir. Para poder simular lo que podría ser un escenario real se dispone en este ejemplo de los siguientes participantes (*Ver Figura 4.19*):

- Servidor
- Usuario plugin *Vuze* 1
- Usuario plugin *Vuze* 2
- Usuario plugin *Vuze* 3

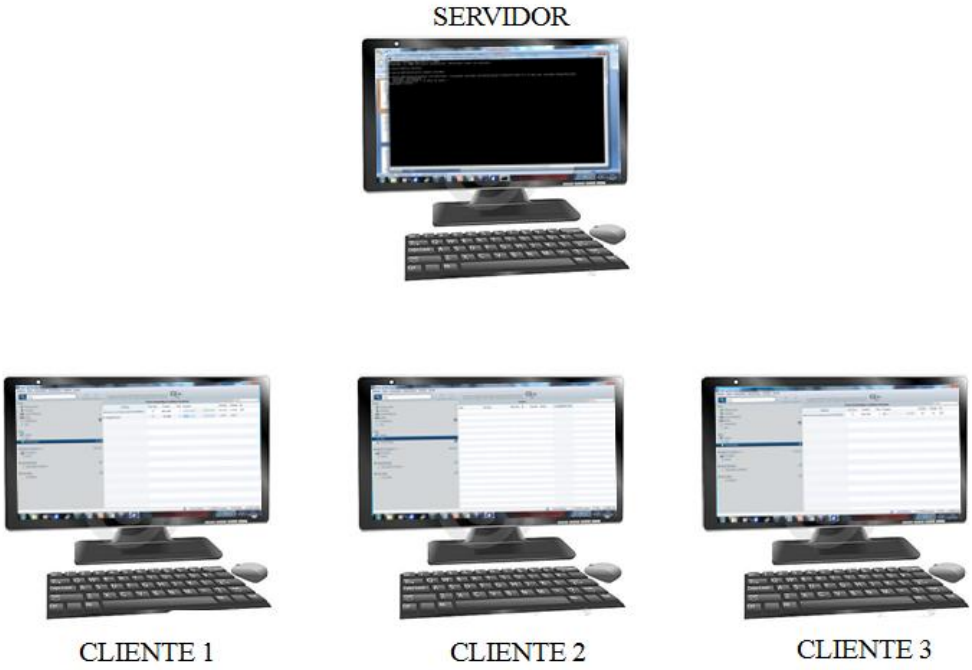


Figura 4.19. Miembros que forman el escenario de ejemplo

Los tres usuarios llevarán a cabo las operaciones pertinentes para realizar las pruebas a la aplicación que corresponden a las siguientes operaciones:

- Instalación plugin: Se determina el resultado de la instalación del plugin en cada usuario en función de las descargas que poseen. Se contemplan que podrían existir en la librería descargas iniciadas con anterioridad a la instalación.
- Desinstalación plugin : No conlleva ninguna prueba en particular.
- Reinstalación plugin: Se atiende a la posibilidad de que en el período en el que el plugin ha estado desinstalado haya habido variaciones en su librería (nuevas descargas o cambios en el estado de las ya existentes) por lo que se compara ésta con la base de datos para su actualización.
- Nueva descarga: Se tratan las dos posibilidades que podrían ocurrir:
  - Descarga existente para otro usuario en la base de datos.
  - Descarga inexistente en la base de datos.
- Cambio de estado en una descarga: Se tratan los dos motivos de cambio en el estado, bien por la cancelación de una descarga o porque se haya completado.

Preparación aplicación

En primer lugar, se recurre por parte del servidor al *ANEXO 1. Manual de puesta en marcha del servidor* para arrancarlo. En este momento la situación actual es:

- Base de datos con las tablas vacías.
- Servidor a la espera de recibir clientes.

Llegada a esta situación el servidor muestra el mensaje de la *Figura 4.20* indicando que está preparado para comunicarse con los clientes que instalen el plugin.

```
-- Servidor configurado --
-- Servidor conectado a la base de datos --
Esperando cliente...
```

Figura 4.20. Mensaje mostrado al arrancar el servidor

Instalación Cliente 1

Siguiendo los pasos indicados en el *Anexo 2. Manual del plugin para el usuario*, el Cliente 1 completa la instalación. Inicia la comunicación con el servidor por primera vez lo que implica que se le asigna un identificador de usuario.

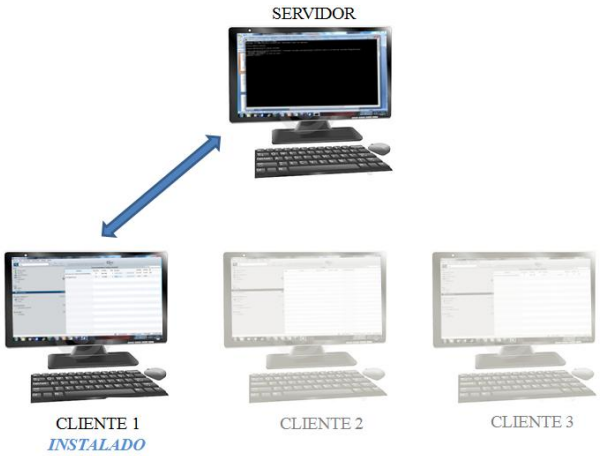


Figura 4.21. Situación de ejemplo con el Cliente 1 instalado.

El contexto en el que el Cliente 1 instala el plugin es con dos descargas en su librería, una de ellas descargada completamente con anterioridad y la otra en proceso de descarga. Con los datos recibidos la base de datos presenta el contenido mostrado en la *Figura 4.22*.

Tabla *usuarios*

id_usuario
45453587729043333620

Tabla *sesiones*

id_usuario	conexion	desconexion
45453587729043333620	Sat Mar 26 13:41:39 CET 2011	

Tabla *descargas*

id_torrent	nombre_fichero	nombre_torrent
367fdb497f7f8a19981475476c8d6e40166c8164	Picasa.Photo.Editor.2.0.WorkSuite	367FDB497F7F8A199881475476C8D6E40166C8164.torrent
5d85099ab65c18fbf61e5e5412572c47795c6a19	AVG Anti-Virus Professional 9.0 Build 663a1706 + Keygen [RH]	5D85099AB65C18FBF61E5E5412572C47795C6A19.torrent

Tabla *realiza*

id_usuario	id_torrent	inicio	fin	estado
45453587729043333620	367fdb497f7f8a19971475476c8d6e40166c8164	INICIO_DESCONOCIDO		0
45453587729043333620	5d85099ab65c18fbf61e5e5412572c47795c6a19	INICIO_DESCONOCIDO	FIN_DESCONOCIDO	1

Figura 4.22. Tablas de la base de datos tras la instalación del Cliente 1

Los campos de timestamp *inicio* y *fin* que figuran como INICIO_DESCONOCIDO o FIN_DESCONOCIDO se deben a que esas descargas han comenzado o finalizado antes de que el plugin fuera instalado por lo que se desconocen dichos valores.

Variación de los estados de las descargas del Cliente 1

El estado del contenido que se encontraba en estado de descarga para el Cliente 1 varía a *Completo*. De igual modo, el Cliente 1 decide cancelar la descarga que ya había finalizado eliminándola de su librería. Al mantenerse instalado el plugin en este Cliente, el servidor es avisado de la actualización de ambas descargas y las lleva a cabo en tiempo real en la base de datos.

Tabla *realiza*

id_usuario	id_torrent	inicio	fin	estado
45453587729043333620	367fdb497f7f8a19971475476c8d6e40166c8164	INICIO_DESCONOCIDO	Sat Mar 26 14:30:09 CET 2011	1
45453587729043333620	5d85099ab65c18fbf61e5e5412572c47795c6a19	INICIO_DESCONOCIDO	FIN_DESCONOCIDO	2

completo

cancelado

Figura 4.23. Modificación del estado de una descarga

Instalación Cliente 2

Posteriormente, el Cliente 2 instala el plugin con la diferencia de que no cuenta con ninguna descarga en su librería.

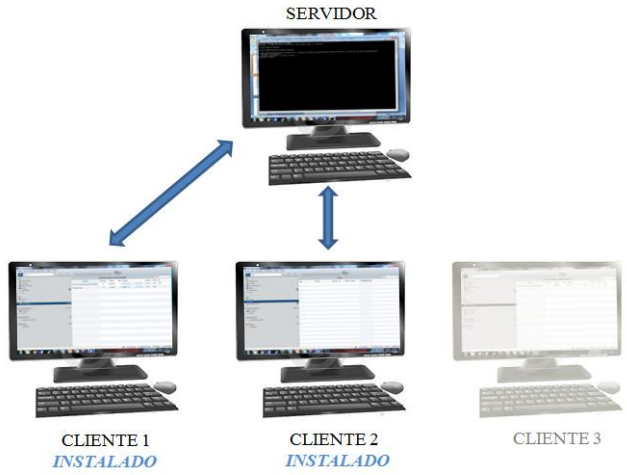


Figura 4.24. Estado de los participantes al instalar el Cliente 2

En este momento se modifican la tabla *usuarios* y la tabla *sesiones* para actualizarlas con la información del Cliente 2. El resto de tablas no se ven alteradas puesto que el nuevo cliente no cuenta con descargas.

Tabla *usuarios*

	id_usuario
Cliente 1	45453587729043333620
Cliente 2	53768000651549592969

Tabla *sesiones*

id_usuario	conexion	desconexion
45453587729043333620	Sat Mar 26 13:41:39 CET 2011	
53768000651549592969	Sat Mar 26 14:19:00 CET 2011	

Figura 4.25. Contenido de las tablas modificadas tras la instalación del Cliente 2

Nueva descarga para el Cliente 2

Si el cliente 2 decide añadir una descarga a su librería el servidor será avisado de la novedad que añadirá a la base de datos. Al tratarse de un contenido no registrado anteriormente para ningún cliente se almacenan los datos tanto de la tabla *descargas* como de la tabla *realiza*.

Tabla *descargas*

id_torrent	nombre_fichero	nombre_torrent
367fdb497f7f8a19981475476c8d6e40166c8164	Picasa.Photo.Editor.2.0.WorkSuite	367FDB497F7F8A199881475476C8D6E40166C8164.torrent
5d85099ab65c18fbf61e5e5412572c47795c6a19	AVG Anti-Virus Professional 9.0 Build 663a1706 + Keygen [RH]	5D85099AB65C18FBF61E5E5412572C47795C6A19.torrent
7a0dbdfc37f3bcc051d9c89d160eb8c469381d48	Mysql Reference Manual – allfreebook.tk	7A0DBDFC37F3BCC051D9C89D160EB8C469381D48.torrent

Tabla *realiza*

id_usuario	id_torrent	inicio	fin	estado
45453587729043333620	367fdb497f7f8a19971475476c8d6e40166c8164	INICIO_DESCONOCIDO	Sat Mar 26 14:30:09 CET 2011	1
45453587729043333620	5d85099ab65c18fbf61e5e5412572c47795c6a19	INICIO_DESCONOCIDO	FIN_DESCONOCIDO	2
53768000651549592969	7a0dbdfc37f3bcc051d9c89d160eb8c469381d48	Sat Mar 26 14:53:35 SAT 2011		0

Nueva descarga Cliente 2

Figura 4.26. Reflejo de la nueva descarga del Cliente 2 en la base de datos

Desinstalación Cliente 1

El Cliente 1 decide desinstalar el plugin. El servidor al recibir la notificación que se lo indica modifica el estado de su sesión en la tabla *sesiones*.

Figura 4.27. Estado de los clientes tras la desinstalación del Cliente 1

Tabla *sesiones*

id_usuario	conexion	desconexion
45453587729043333620	Sat Mar 26 13:41:39 CET 2011	Sat Mar 26 15:10:10 CET 2011
53768000651549592969	Sat Mar 26 14:19:00 CET 2011	

Desinstalación plugin Cliente 1

Figura 4.28. Inserción del timestamp de desinstalación en el Cliente 1

Instalación Cliente 3

El Cliente 3 procede a instalar el plugin contando en su librería con un archivo que está descargándose y que se trata del mismo que el Cliente 1 descargó en su momento.

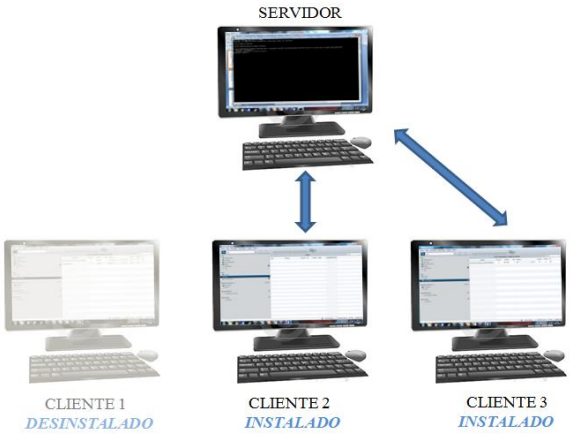


Figura 4.29. Estado actual de los clientes con la instalación del Cliente 3

Dado que la descarga con la que cuenta el Cliente 3 es conocida por el servidor que ya la registró para el Cliente 1 (existe ya en la tabla *descargas*), sólo se añade el registro oportuno en la tabla *realiza* tal y como indica la *Figura 4.30*.

Tabla *usuarios*

	id_usuario
Cliente 1	45453587729043333620
Cliente 2	53768000651549592969
Cliente 3	77229939211205892013

Tabla *sesiones*

id_usuario	conexion	desconexion
45453587729043333620	Sat Mar 26 13:41:39 CET 2011	Sat Mar 26 15:10:10 CET 2011
53768000651549592969	Sat Mar 26 14:19:00 CET 2011	
77229939211205892013	Sat Mar 26 16:40:21 CET 2011	

Tabla *descargas*

id_torrent	nombre_fichero	nombre_torrent
367fdb497f7f8a19981475476c8d6e40166c8164	Picasa.Photo.Editor.2.0.WorkSuite	367FDB497F7F8A199881475476C8D6E40166C8164.torrent
5d85099ab65c18fbf61e5e5412572c47795c6a19	AVG Anti-Virus Professional 9.0 Build 663a1706 + Keygen [RH]	5D85099AB65C18FBF61E5E5412572C47795C6A19.torrent
7a0dbdfc37f3bcc051d9c89d160eb8c469381d48	Mysql Reference Manual – allfreebook.tk	7A0DBDFC37F3BCC051D9C89D160EB8C469381D48.torrent

Cliente 1

Tabla *realiza*

id_usuario	id_torrent	inicio	fin	estado
45453587729043333620	367fdb497f7f8a19971475476c8d6e40166c8164	INICIO_DESCONOCIDO	Sat Mar 26 14:30:09 CET 2011	1
45453587729043333620	5d85099ab65c18fbf61e5e5412572c47795c6a19	INICIO_DESCONOCIDO	FIN_DESCONOCIDO	2
53768000651549592969	7a0dbdfc37f3bcc051d9c89d160eb8c469381d48	Sat Mar 26 14:53:35 SAT 2011		0
77229939211205892013	367fdb497f7f8a19971475476c8d6e40166c8164	INICIO_DESCONOCIDO		0

Cliente 2

Cliente 3

Figura 4.30. Contenido de las tablas añadiendo una descarga ya existente para otro usuario

Nueva descarga en el Cliente 1

El Cliente 1 procede a incorporar una nueva descarga a su librería. Como en este instante no cuenta con el plugin instalado el servidor no se percata de esta novedad y, por lo tanto, no se produce ningún cambio en los datos almacenados.

Reinstalación Cliente 1

El Cliente 1 procede a instalar el plugin de nuevo. El proceso de reinstalación en un usuario conlleva la comprobación de las descargas de la librería verificando las novedades en las mismas. Estas novedades implican inserciones de nuevas descargas en la base de datos o modificaciones en el estado de las ya existentes. Esto ocurre porque en el tiempo en el que el plugin ha estado desinstalado han podido variar las descargas de su librería, como es el caso de este ejemplo.

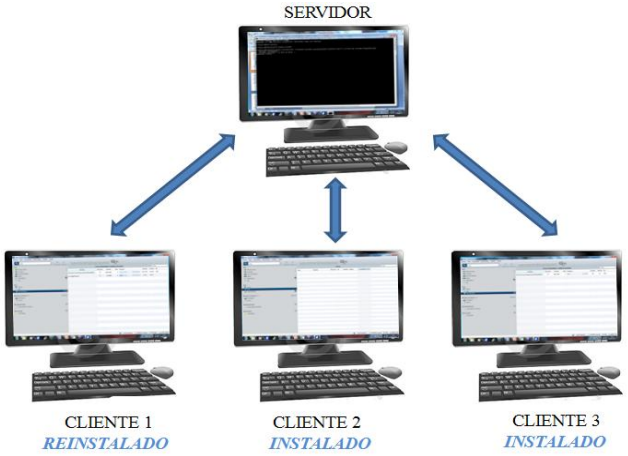


Figura 4.31. Escenario tras la reinstalación del Cliente 1

Tabla *sesiones*

id_usuario	conexion	desconexion
45453587729043333620	Sat Mar 26 13:41:39 CET 2011	Sat Mar 26 15:10:10 CET 2011
53768000651549592969	Sat Mar 26 14:19:00 CET 2011	
77229939211205892013	Sat Mar 26 16:40:21 CET 2011	
45453587729043333620	Sat Mar 26 17:00:45 CET 2011	

Tabla *descargas*

id_torrent	nombre_fichero	nombre_torrent
367fdb497f7f8a19981475476c8d6e40166c8164	Picasa.Photo.Editor.2.0.WorkSuite	367FDB497F7F8A199881475476C8D6E40166C8164.torrent
5d85099ab65c18fbf61e5e5412572c47795c6a19	AVG Anti-Virus Professional 9.0 Build 663a1706 + Keygen [RH]	5D85099AB65C18FBF61E5E5412572C47795C6A19.torrent
7a0dbdfc37f3bcc051d9c89d160eb8c469381d48	Mysql Reference Manual – allfreebook.tk	7A0DBDFC37F3BCC051D9C89D160EB8C469381D48.torrent
b5670bec1c6ae53a87d6dc1c55916cd9858a2c1c	Tchaikovsky – The Nutcracker [Complete]	Tchaikovsky – The Nutcracker [Complete].torrent

Tabla *realiza*

id_usuario	id_torrent	inicio	fin	estado
45453587729043333620	367fdb497f7f8a19971475476c8d6e40166c8164	INICIO_DESCONOCIDO	Sat Mar 26 14:30:09 CET 2011	1
45453587729043333620	5d85099ab65c18fbf61e5e5412572c47795c6a19	INICIO_DESCONOCIDO	FIN_DESCONOCIDO	2
53768000651549592969	7a0dbdfc37f3bcc051d9c89d160eb8c469381d48	Sat Mar 26 14:53:35 SAT 2011		0
77229939211205892013	367fdb497f7f8a19971475476c8d6e40166c8164	INICIO_DESCONOCIDO		0
45453587729043333620	b5670bec1c6ae53a87d6dc1c55916cd9858a2c1c	INICIO_DESCONOCIDO	FIN_DESCONOCIDO	1

→ Nueva descarga añadida por la reinstalación del plugin en el Cliente 1

Figura 4.32. Tablas afectadas por la reinstalación en el Cliente 1

4.6. ESTRUCTURA DE DIRECTORIOS

La aplicación completa se localiza en dos carpetas referentes a los dos módulos implementados: PLUGIN y SERVIDOR. En ambas figuran los ficheros *.project* y *.classpath* que son archivos de configuración generados por *Eclipse*.

A continuación se especifica la estructura de subdirectorios de las dos carpetas y la descripción de los archivos con los que cuentan:

► PLUGIN

Tal y como indica la *Figura 4.33* esta carpeta cuenta con el archivo empaquetado del plugin para instalar desde Vuze(*plugin_vuze_0.1.jar*) y de un directorio (*plugin_vuze*) que guarda el proyecto del plugin.

Las clases que lo componen son:

- *PluginVuze.java:* Es la clase principal que se encarga de la inicialización del plugin, de la respuesta ante eventos de instalación, desinstalación y cambios en el estado de las descargas, de obtener las descargas del usuario, del envío de la información recogida al servidor abriendo una comunicación, de los avisos mostrados al usuario, de la comunicación con la aplicación *FakeTorrent* y de la página de configuración]
- *ManejadorFichero.java:* Clase responsable de las operaciones necesarias en el tratamiento del fichero de texto que guarda el identificador de usuario. La funcionalidad de esta clase reside en permitir que el programa principal pueda consultar si existe un fichero de texto que almacena cada cliente con el identificador que le caracteriza, así como de llevar a cabo la generación de dicho identificador y su escritura en un fichero.
- *CompruebaHash.java:* Clase que obtiene los resultados de la aplicación FakeTorrent determinando a partir de un identificador de torrent si corresponde o no a un contenido falso.

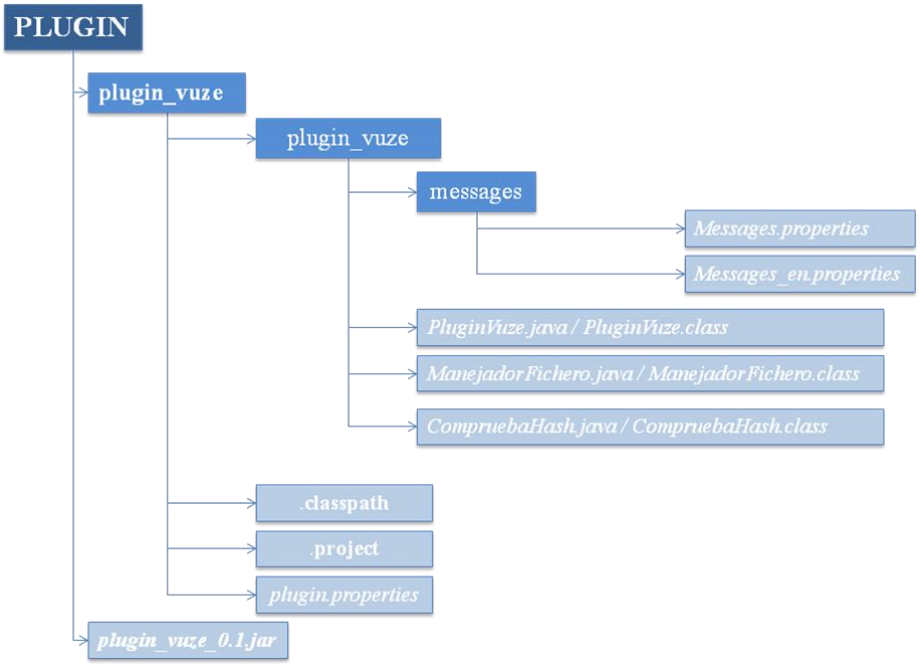


Figura 4.33. Organización de los directorios del plugin

Existe un fichero de propiedades llamado *plugin.properties* donde se definen propiedades que caracterizan al plugin y es necesario para que Vuze encuentre el plugin.

► SERVIDOR

La carpeta SERVIDOR cuenta con el directorio que contiene el proyecto del servidor, el fichero con las tablas de la base de datos (*datos_plugin.sql*) y la librería del conector JDBC (*mysql-connector-java-5.1.13-bin.jar*). Las clases que lo componen son:

- *PluginServidor.java:* Servidor de la aplicación encargado de recoger los datos de los usuarios del plugin y almacenarlos en la base de datos. Para ello instancia un socket servidor que acepta las conexiones de los clientes recibiendo los parámetros de sus descargas y sirviendo como conector con la base de datos a la cual envía la información oportuna.
- *ManejadorBBDD.java*: Realiza la conexión a la base de datos y las operaciones que el servidor realiza sobre la misma. Contiene los métodos utilizados por el servidor para poder conectarse a la base de datos y para poder realizar operaciones de inserción, consulta y actualización de información.

Existe un fichero en el proyecto del servidor denominado c*onfig_serv.properties* que trata de un archivo de configuración de los parámetros requeridos por el servidor. Sería necesaria su modificación si alguno de ellos requiere ser cambiado: nombre, usuario y contraseña de la base de datos y puerto del servidor.

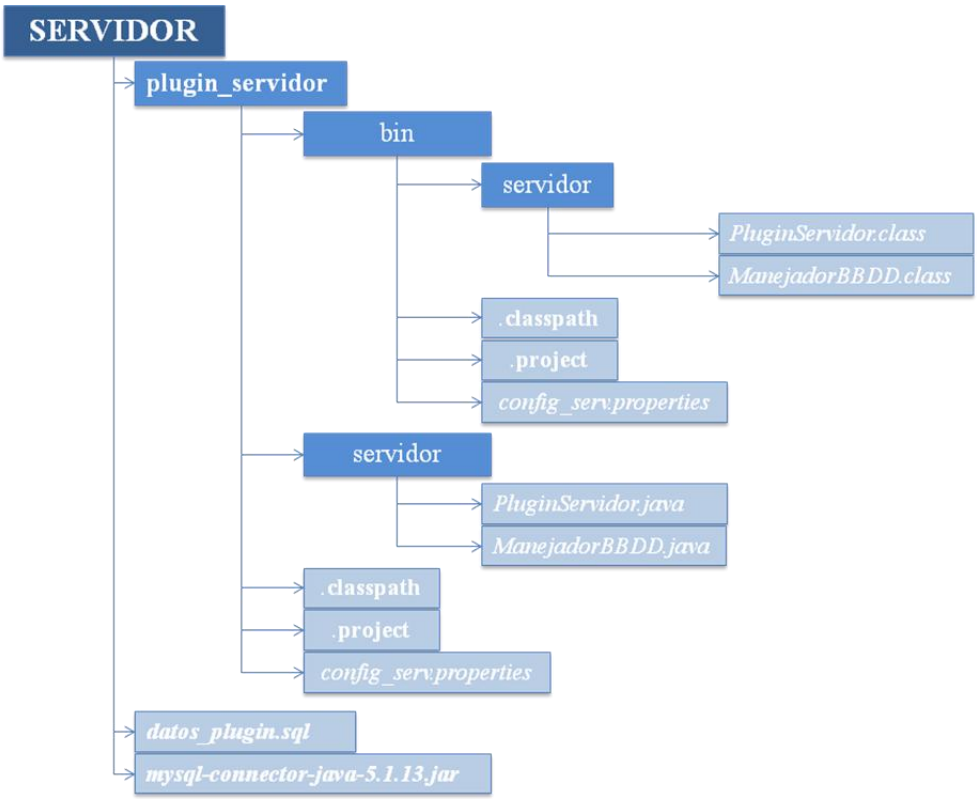


Figura 4.34. Organización de los directorios del servidor

5. HERRAMIENTAS

En este apartado se describen los lenguajes y herramientas utilizados en el diseño e implementación de la aplicación, así como las versiones utilizadas.

Figura 5.1. Herramientas empleadas en el desarrollo del Proyecto

5.1. ENTORNO DE DESARROLLO: Eclipse

Para hacer más cómoda la programación existen los IDEs (*Integrated Development Environment*). Los entornos de desarrollo integrados permiten desarrollar las aplicaciones de forma mucha más rápida, incorporando en muchos casos librerías con componentes ya desarrollados, los cuales se añaden al proyecto o programa. Los más extendidos son *Eclipse* y *NetBeans*. Ambos entornos permiten programación de aplicaciones de consola, de aplicaciones web y de aplicaciones visuales.

Se ha empleado *Eclipse* [32] [33] [34] para el desarrollo de la aplicación. Se trata de un entorno de desarrollo integrado de código abierto multiplataforma. Fue desarrollado originalmente por IBM para reemplazar a *VisualAge*. Su desarrollador pasó a ser en 2003 la Fundación Eclipse, una organización independiente de IBM sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse desarrolla "Aplicaciones de Cliente Enriquecido", lo opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Los siguientes componentes constituyen la plataforma de cliente enriquecido que son la base de *Eclipse*:

- *Plataforma principal*: inicio de Eclipse, ejecución de plugins.
- *OSGi*: una plataforma para bundling estándar.
- El *Standard Widget Toolkit* (SWT): Un widget toolkit portable.
- *JFace*: manejo de archivos, manejo de texto, editores de texto.
- El *Workbench* de *Eclipse*: vistas, editores, perspectivas, asistentes.

La elección de *Eclipse* como entorno de desarrollo para este Proyecto es debida a las facilidades que aporta en cuanto a edición, compilación y ejecución de programas Java durante su fase de desarrollo. Según la herramienta de cómputo de líneas de código fuente *SLOCCount* se demostró que Java es el lenguaje más utilizado en *Eclipse*.

Por otro lado, es recomendado para el desarrollo de aplicaciones cliente como *Vuze* por lo que ofrece las características necesarias para ser el entorno adecuado en este Proyecto.

La versión de Eclipse utilizada es la **3.6.** disponible para su descarga en:
http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.6-201006080911/eclipse-SDK-3.6-win32.zip

5.2. LENGUAJE DE PROGRAMACIÓN: Java

Java [35] es un lenguaje de programación orientado a objetos diseñado por Sun Mycrosystems y perteneciente en la actualidad a Oracle. La compañía Sun lo describe como un lenguaje "simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico".

El código se puede escribir en cualquier editor de texto y para compilar el código en bytecodes hace falta descargar gratuitamente la versión del *Java Development Kit* (JDK) adecuada. Este software consta de las herramientas de desarrollo para la creación de programas en Java. La versión del JDK utilizada en el desarrollo de la aplicación es **1.6.0_21**.

Para poder ejecutar programas Java es necesario el *Java Runtime Environment* (JRE) (que está incluido en el JDK). Este conjunto de utilidades actúa como un intermediario entre el sistema operativo y Java.

La elección de Java como lenguaje de programación en el desarrollo de este Proyecto está basada en las ventajas que este lenguaje aporta en cuanto a la independencia de la plataforma que lo ejecute y a que *Vuze*, programa para el que está implementado el plugin, dispone de una interfaz de programación de aplicaciones (*API*) para el desarrollo de plugins en Java.

Los paquetes de Java que se han requerido en la implementación del programa son:

- `java.io`
  Se ha utilizado para la entrada y salida a través de flujos de datos, y para el tratamiento de ficheros.
- `java.net`
  Contiene las clases que se han necesitado para llevar a cabo la comunicación mediante sockets.
- `java.util`
  Las clases de este paquete se han requerido para la generación aleatoria de números y para la obtención de timestamp.

5.3. CLIENTE BITTORRENT: Vuze

Vuze [15] [16] es el cliente BitTorrent utilizado en el Proyecto y para el cual se ha desarrollado el plugin. Se detallan aspectos como su interfaz, funcionamiento o características en el apartado 3.2. así como un manual de instalación del plugin en el *Anexo 3. Manual del plugin para el usuario*.

Los motivos por los que se ha escogido *Vuze* para la implementación del plugin son los siguientes:

- Es uno de los clientes BitTorrent más utilizados en la actualidad, por lo que ampliar sus características mejora sus prestaciones y amplía sus ventajas frente a otros clientes.
- Es de código abierto.

- Está desarrollado en lenguaje Java y dispone de una API que ofrece gran cantidad de paquetes específicos para el desarrollo de plugins en *Vuze* (*http://cdn01.vuze.com/site/dev/javadoc/Vuze_4502/*), entre los que destacan:
  - `org.gudy.azureus2.plugins.Plugin`
    - Define la interfaz del plugin.
  - `org.gudy.azureus2.plugins.download.DownloadManager`
    - Da acceso a las funciones utilizadas para monitorizar y administrar las descargas de *Vuze*.
  - `org.gudy.azureus2.plugins.torrent.Torrent`
    - Ofrece gran cantidad de métodos que permiten obtener los diversos parámetros de los torrents.
  - `org.gudy.azureus2.plugins.download.Download`
    - Ofrece gran cantidad de métodos que permiten obtener los diversos parámetros de las descargas.
  - `org.gudy.azureus2.plugins.ui.model.BasicPluginConfigModel`
    - Permite diseñar una sección de configuración del plugin.
  - `org.gudy.azureus2.plugins.logging.LoggerChannel`
    - Implementa los métodos que manejan el canal de registro de un plugin utilizado para avisos informativos o depuración.

La versión de *Vuze* sobre la que se ha trabajado para el desarrollo del Proyecto es la **4.6** disponible en: *http://www.vuze.com*

5.4. BASE DE DATOS: MySQL

MySQL [36] es un Sistema Gestor de Bases de Datos (SGBD) desarrollado en C y C++. Proporciona un servidor de base de datos relacional SQL muy rápido, multihilo y multiusuario. Se ha empleado para el manejo y creación del almacén de datos de las descargas de usuarios.

MySQL es actualmente uno de los SGBD más extendidos y utilizados. Su éxito reside en que no depende de la plataforma, existiendo versiones para distintos sistemas operativos. Entre las múltiples plataformas para las que funciona están: Linux, Microsoft Windows, MAC OS X o Solaris. Otra de las ventajas que posee es que es software libre para usos con licencia GNU (si bien las empresas que deseen la incorporación en productos privativos deben comprar una licencia que les permita ese uso) y soporta múltiples accesos simultáneos.

Las razones fundamentales por las que se ha escogido MySQL como sistema de gestión de bases de datos son los siguientes: es multiplataforma, su uso es bajo licencia GNU y puede conectarse con la aplicación Java a partir del conector JDBC.

La versión de MySQL utilizada en la aplicación es la **5.1.54** disponible para su descarga en: *http://dev.mysql.com/downloads/mysql/5.1.html*

5.5. CONECTOR: JDBC

Java Database Connectivity (JDBC) [37] [38] [39] es una interfaz de programación de aplicaciones (API) creada por SUN Microsystems como parte del JDK 1.1. Esta especificación es un conjunto de clases y métodos de operación que permiten a cualquier programa Java acceder a sistemas de bases de datos. La ejecución de operaciones sobre bases de datos desde el lenguaje Java es independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede.

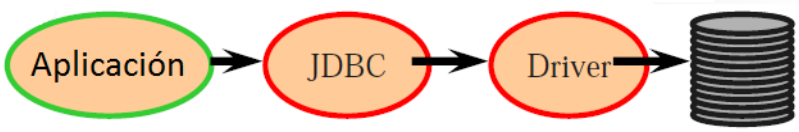


Figura 5.2. Esquema de los recursos empleados para comunicar la aplicación con la base de datos

Para utilizar la base de datos desde el servidor Java, éste se ejecuta junto con la biblioteca de conexión apropiada al modelo de su base de datos, y se accede a ella estableciendo una conexión. Para ello provee el localizador a la base de datos y los parámetros de conexión específicos (ruta a la base de datos según el esquema: "*jdbc: subprotocolo:subnombre*", usuario y contraseña).

A partir de ese momento la conexión está establecida y se puede realizar cualquier tipo de acción con la base de datos para las que se disponga de permiso:

- ✓ Consulta de datos.
- ✓ Actualización de datos.
- ✓ Creación, modificación y borrado de tablas.
- ✓ Ejecución de procedimientos almacenados en la base de datos.

El paquete que proporciona el acceso y procesamiento de los datos almacenados en una base de datos empleando Java es *java.sql*. La gestión de los controladores JDBC es proporcionada por la clase *DriverManager* de dicho paquete.

La versión instalada del conector JDBC es la **5.1.13** cuya descarga está disponible en el enlace: *http://www.mysql.com/downloads/connector/j/*

6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1. CONCLUSIONES

Tal y como se ha especificado anteriormente, la aplicación desarrollada forma parte de un trabajo más extenso del *Departamento de Ingeniería Telemática*. La finalidad de este Proyecto era implementar un plugin para los usuarios de *Vuze* que quieran ampliar la funcionalidad de su cliente BitTorrent con un sistema de recomendación de contenidos y detección de contenidos falsos.

Al inicio de este Proyecto se plantearon dos objetivos principales que servirán de base para el desarrollo del futuro sistema:

- Almacenar información detallada de las descargas existentes en la librería de los usuarios de *Vuze*.
- Proveer de un sistema de alerta que avise al usuario de la falsedad del contenido de sus descargas.

Ambos objetivos se han cumplido al conseguir recoger parámetros relativos a las descargas, como los identificadores de torrent o los nombres de los ficheros, y la relación de las descargas con los usuarios que las realizan. Toda esta información se recoge en una base de datos continuamente actualizada que facilitará su posterior tratamiento en el desarrollo del sistema completo en el que se enmarca el Proyecto. Por otro lado, se ha provisto de un sistema de alerta que mediante avisos en *Vuze* indica al usuario si va a descargar un fichero cuyo contenido no corresponde al esperado a partir de los resultados aportados por la aplicación *FakeTorrent* desarrollada en el *Departamento de Ingeniería Telemática*.

Para lograr cumplir los requerimientos marcados, este Proyecto ha desarrollado una aplicación compuesta por un plugin para los usuarios de *Vuze* que recoge los datos de sus descargas y un programa servidor al que se envían dichos datos para ser almacenados.

A partir del historial de resultados que ofrece este Proyecto se desarrollará un sistema de recomendación de contenidos basado en filtrado colaborativo que ampliará la funcionalidad de *Vuze* ofreciendo facilidades a sus usuarios. La simplificación del proceso que tienen que llevar a cabo los usuarios es la principal ventaja de este sistema. Por otro lado, el sistema dispone de un mecanismo de alerta de contenido falso basado en los perfiles de los usuarios que suben este tipo de contenido.

Una de las particularidades destacables del programa es el archivo del que debe disponer el usuario que consta de un único fichero de un tamaño muy pequeño y de fácil instalación. Esta característica simplifica la operación que tiene que llevar a cabo el usuario y facilita la difusión del plugin.

Reducir el tiempo de búsqueda de torrents que puedan interesarles al recibir recomendaciones personalizadas y ser avisados de descargas de archivos falsos hace que el proyecto completo pueda resultar de interés para un gran número de usuarios. Por lo que la aportación que el sistema hará en un cliente tan popular como *Vuze* le añade un

valor de gran importancia en este tipo de aplicaciones que resulta muy atractivo a los usuarios. Dado que el objetivo del proyecto completo desarrollado en el Departamento persigue mejorar las prestaciones de los clientes BitTorrent actuales, la aplicación desarrollada en este Proyecto podría extenderse a otros clientes siendo adaptada a los mismos para poder mejorar también sus prestaciones.

Existen posibles mejoras que podrían dar una solución que añadiera cierta complejidad pero que convertiría la aplicación en una más completa. Entre ellas están:

- Obtención de más parámetros de los torrents (como un listado de trackers con el contenido) o de las descargas (como el tamaño del fichero) que permitan que la recomendación de contenidos se ajuste a otros aspectos adicionales.
- Guardar la relación de descargas para los usuarios en cada una de sus sesiones, lo que permitiría disponer de información más precisa.
- Generar un fichero en la máquina del usuario donde queden registradas las sesiones que ha tenido éste en *Vuze* mientras que el plugin ha estado instalado. Así se podría comprobar de una manera más concreta si el cliente ha tenido instalado el plugin en alguna ocasión anterior.
- Proporcionar la posibilidad de descartar automáticamente las descargas con contenido falso o realizarle una consulta al usuario.

Características como las ofrecidas en este plugin presentan a *Vuze* como un programa actualizado y adaptado a los nuevos requerimientos. El contexto actual hace que pueda hacer frente a otros clientes BitTorrent que ofrecen funcionalidades orientadas en la misma línea e incluso a la aparición de nuevos clientes que ofrecen descentralización de las descargas y recomendación de contenidos pero carecen de la popularidad de *Vuze*.

6.2. TRABAJOS FUTUROS

Dado el marco en el que está orientado el Proyecto, las futuras líneas de desarrollo van ligadas al sistema completo de mejora de prestaciones que automatizará el procedimiento llevado a cabo por los usuarios basado en la información recogida por el plugin.

Sin embargo, la parte del sistema completo que se ha desarrollado en este Proyecto ofrece otras posibilidades en cuanto a futuras líneas de investigación independientes del sistema de recomendación para el cual formará parte. El hecho de que se proporcione una base de datos que va actualizándose en tiempo real con las descargas de los usuarios que tengan instalado el plugin puede enfocarse a otros ámbitos como:

- *Publicidad*: En función de qué tipo de archivos descarga cada usuario se pueden determinar sus posibles intereses obteniendo distintos perfiles y recurriendo así al marketing personalizado.

- *Análisis del comportamiento de los usuarios*: Pueden obtenerse patrones de comportamiento de los usuarios a partir de la duración media de las sesiones o de la frecuencia de conexiones.

- *Estudios estadísticos*: A partir de los datos recogidos pueden obtenerse estadísticas sobre, por ejemplo, el tiempo medio en el que tardan las descargas en completarse o los períodos de mayor actividad.

- *Rankings de descargas*: Con la finalidad de elaborar un listado de descargas con las más populares para ofrecérselas al usuario a partir de contabilizar cuáles son las más comunes a los usuarios del plugin.

REFERENCIAS

[1] Fundación Orange. 2010 eEspaña: *Informe Anual sobre el Desarrollo de la Sociedad de la Información en España*.
http://www.informeeespana.es/docs/eE2010.pdf

[2] Observatorio Nacional de las Telecomunicaciones y la Sociedad de la Información (ONTSI). *La Sociedad en Red 2009*. Informe Anual, edición 2010.
http://www.ontsi.red.es/informes-anuales/articles/id/4814/informe-anual-2009-edicion-2010.html

[3] BitTorrent Blog. *100 Million and Growing!* Enero 2011
http://blog.bittorrent.com/2011/01/03/100-million-and-growing/

[4] Bernardo Quintero. *P2P Distribuido y seguro*. Hispasec Enero 2010.
http://www.hispasec.com/unaaldia/4097/

[5] Marios Iliofotou, Georgos Siganos, Xiaoyuan Yang, Pablo Rodríguez. *Comparing BitTorrent Clients in the Wild: The Case of Download Speed*. IPTPS, 2010.
http://www.usenix.org/event/iptps10/tech/slides/iliofotou.pdf

[6] Sergio Manuel Galán Nieto. *Filtrado Colaborativo y Sistemas de Recomendación*. Universidad Carlos III de Madrid, 2007.
http://www.it.uc3m.es/jvillena/irc/practicas/06-07/31.pdf

[7] Daniela Godoy. *Sistemas de Recomendación de Información*. UNICEN University, Argentina.
http://www.exa.unicen.edu.ar/catedras/ageinweb/slides/intro-1p.pdf

[8] Vertor. Verified Torrents.
http://www.vertor.com

[9] Rubén Cuevas, Michal Kryczka, Ángel Cuevas, Sebastian Kaune, Carmen Guerrero y Reza Rejaie. *Is content Publishing in BitTorrent Altruistic or Profit-Driven?* ACM CoNEXT 2010.
http://conferences.sigcomm.org/co-next/2010/CoNEXT_papers/11-Cuevas.pdf

[10] Bram Cohen, *The BitTorrent Protocol Specification*
http://www.bittorrent.org/beps/bep_0003.html

[11] Alejandro Pardo Tapia, Diego Martínez Campos. *Protocolo BitTorrent*. Universidad Técnica Federico Santa María.
http://profesores.elo.utfsm.cl/~agv/elo322/1s09/project/reports/proyecto_bittorrent_elo322.pdf

[12] *Documentación BitTorrent*.
http://www.doc.ubuntu-es.org

[13] *Manual BitTorrent*.
http://www.ayudabittorrent.com

[14] Equipo de investigación Tribler. *uTorrent Dominates BitTorrent Client Market Share*. Universidad Tecnológica de Delft, Holanda. 2009
http://torrentfreak.com/utorrent-dominates-bittorrent-client-market-share-090624/

[15] *Página Oficial de Vuze*.
http://www.vuze.com

[16] *Vuze Wiki*.
http://wiki.vuze.com

[17] *Artículo Wikipedia Vuze*.
http://es.wikipedia.org/wiki/Vuze

[18] *Vuze -Vuze HD*.
http://vuze.pro

[19] *Página Oficial de µTorrent*.
http://www.utorrent.com

[20] *µTorrent*.
http://utorrent.pro/

[21] *Página Oficial de BitTorrent.*
http://www.bittorrent.com

[22] *BitComet Help Wiki*.
http://wiki.bitcomet.com/

[23] *Página Oficial de Transmission.*
http://www.transmissionbt.com/

[24] *Artículo Wikipedia Transmission*.
http://es.wikipedia.org/wiki/Transmission_(BitTorrent)

[25] T*ribler*.
http://www.tribler.org

[26] *Chrysalis*.
http://www.bittorrent.com/chrysalis/

[27] *BitMate*.
http://www.dritte.org/bitmate.html

[28] John O'Conner. *Java Internationalization: Localization with ResourceBundles.* Octubre 1998.
http://java.sun.com/developer/technicalArticles/Intl/ResourceBundles/

[29] Patricia Martínez Barco. *JSockets*. Departamento de Sistemas y Lenguajes Informáticos, Universidad de Alicante.
http://www.dlsi.ua.es/asignaturas/sid/JSockets.pdf

[30] Departamento de Informática, E. U. Informática de Segovia. *Sistemas Distribuidos: Sockets en Java.* Universidad de Valladolid.
http://www.infor.uva.es/~fdiaz/sd/doc/java.net.pdf

[31] *JDBC (TM) Database Access*. The Java™ Tutorials.
http://download.oracle.com/javase/tutorial/jdbc/index.html

[32] *The Eclipse Foundation open source community* website.
http://www.eclipse.org

[33] *Help - Eclipse documentation*.
http://help.eclipse.org/helios/index.jsp

[34] Jairo Chapela Martínez. *Introducción al entorno de desarrollo Eclipse*. Universidad de Vigo, 2007.
http://www-gris.det.uvigo.es/wiki/pub/Main/MiscResources/Manual_Eclipse.pdf

[35] *Java SE Technical Documentation*.
http://download.oracle.com/javase/

[36] *MySQL 5.0. Reference Manual*.
http://downloads.mysql.com/docs/refman-5.0-es.a4.pdf

[37] *JDBC Overview*.
http://www.oracle.com/technetwork/java/overview-141217.html

[38] Juan Gutiérrez Aguado. Curso de Extensión Universitaria *MySQL y Java*. Departamento de Informática, Universidad de Valencia, 2004.
http://www.uv.es/~jgutierr/MySQL_Java/index.html

[39] Eduardo Fernández-Medina Patón. *JDBC Java con Bases de Datos*. Escuela Superior de Informática. Universidad de Castilla-La Mancha.
http://www.vc.ehu.es/jiwotvim/ISOFT2007-2008/Practicas/BloqueV/JDBC-Access-tr.pdf

ANEXOS

ANEXO 1. Presupuesto

Este anexo presenta el cálculo del coste económico total aproximado que está asociado a la realización de este Proyecto. Para ello se tratan los costes de personal y material.

1. COSTES PERSONALES

El Proyecto se ha desarrollado siguiendo un modelo de siete fases con varias tareas en cada una que se detallan a continuación. Se indica el tiempo dedicado en cada fase en semanas, teniendo en cuenta que cada semana de trabajo ha correspondido a 20 horas.

1. Investigación preliminar

Para poder iniciar el Proyecto se llevaron a cabo una serie de tareas que permitieran adquirir los conocimientos y preparar las herramientas necesarias:

- Toma de contacto con el protocolo BitTorrent y los aspectos relacionados con el Proyecto.
- Seguimiento de la guía de creación de plugins para *Vuze*.
- Instalación del entorno de trabajo con las herramientas necesarias.

Duración: 4 semanas

2. Determinación de los requisitos

Esta fase incluye el período de identificación de objetivos del Proyecto y la determinación de las tareas que llevaban a lograrlos.

Duración: 1 semana

3. Diseño del sistema

Una vez determinados los objetivos y requisitos a cumplir por el Proyecto se procedió al diseño del sistema completo que conforma la aplicación:

- Diseño de las tareas que realiza el plugin.
- Diseño del programa servidor y las operaciones que ejecuta sobre la base de datos.
- Estudio del lenguaje SQL.
- Diseño de la estructura de la base de datos.
- Diseño de la comunicación entre módulos.

Duración: 3 semanas

4. Desarrollo del software

Esta fase abarca la implementación del software de los diferentes módulos que se han desarrollado en el Proyecto:

- Plugin para el cliente.
- Programa servidor.
- Base de datos.
- Comunicación entre los módulos.
- Integración con la aplicación que detecta fakes.

Duración: 12 semanas

5. Pruebas al programa

Las pruebas experimentales al programa tuvieron como objetivo determinar que el sistema cumple con las especificaciones. Para ello se emplearon varios clientes que instalaron el plugin y se han planteado posibles situaciones del programa que permitieran diagnosticar fallos. Esta fase abarca también la corrección de los mismos.

Duración: 3 semanas

6. Implantación y evaluación

Las tareas ejecutadas en esta fase son:

- Generación del fichero con las tablas de la base de datos.
- Preparación de los archivos a distribuir junto a los manuales de utilización.
- Evaluación de los objetivos conseguidos.

Duración: 1 semana

7. Desarrollo de la memoria

La última fase del Proyecto engloba el desarrollo de la memoria donde se recogen, entre otros aspectos, las características de cada parte del sistema, las pruebas realizadas o las conclusiones.

Duración: 9 semanas

Duración total: 33 semanas -> 660 horas

	TAREA	DURACIÓN
1	Investigación preliminar	4
2	Determinación de los requisitos	1
3	Diseño del sistema	3
4	Desarrollo del software	12
5	Pruebas al programa	3
6	Implantación y evaluación	1
7	Desarrollo de la memoria	9

Figura A.1. Tabla con la duración de las tareas en semanas

Considerando unos honorarios de 25 euros/ hora para un Ingeniero Técnico, el coste total asociado al desarrollo del Proyecto es de **16500 euros**.

2. COSTES MATERIALES

Para la realización del Proyecto se ha requerido:

- Un ordenador portátil de precio **600 euros**.
- Un servidor central con el que se comunican los usuarios del plugin cuyo precio es de **800 euros**.

3. COSTE ECONÓMICO TOTAL

A partir de los resultados obtenidos para los gastos de cada uno de los aspectos del Proyecto, se obtiene el siguiente coste total.

COSTES	CANTIDAD
PERSONALES	16500 €
MATERIALES	1400 €

Figura A.2. Resumen de los gastos asociados al Proyecto.

El presupuesto total de este Proyecto asciende a la cantidad de **17900 Euros**.

ANEXO 2. Manual de puesta en marcha del servidor

1. INSTALACIÓN HERRAMIENTAS

Para poner en funcionamiento el servidor de la aplicación es necesario antes tener instaladas una serie de herramientas:

- → En primer lugar, instalar *Java Development Kit (JDK)* disponible para su descarga en: *http://www.oracle.com/technetwork/java/javase/downloads/jdk6-jsp-136632.html*

- → Instalar el entorno de desarrollo *Eclipse* que se encuentra disponible en: *http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.6-201006080911/eclipse-SDK-3.6-win32.zip*

- → Por último, proceder a instalar el gestor de bases de datos *MySQL* disponible en *http://www.mysql.com/downloads/mysql* que almacenará la base de datos necesaria para el correcto funcionamiento de la aplicación.

2. IMPORTACIÓN BASE DE DATOS

Llegado a este punto es necesario abrir la consola de MySQL y crear la base de datos "*datos_plugin*":

```
mysql> CREATE DATABASE datos_plugin;
```

A continuación se debe importar la base de datos a partir del archivo "*datos_plugin.sql*" que se facilita junto con la documentación del Proyecto para la creación de las tablas que componen la base de datos. Para ello se debe ejecutar el siguiente comando desde la consola de MySQL (indicando en ruta la referencia al directorio donde se encuentra el archivo *.sql*):

```
mysql> source ruta/datos_plugin.sql
```

3. MODIFICACIÓN PARÁMETROS DE MySQL

Para la correcta comunicación del programa servidor con la base de datos es necesario indicarle al programa servidor los parámetros del gestor de la base de datos con los cuáles se accede.

Para ello, hay que abrir el fichero *config_servidor.properties* que se encuentra en 'plugin_servidor/' y sustituir los parámetros *bbdd.usuario* y *bbdd.contrasena* por los que den el acceso al servidor de MySQL del usuario.

4. GENERACIÓN ARCHIVO .jar

Se abre en el workspace de Eclipse el Proyecto *plugin_servidor* (disponible junto a la documentación del Proyecto) y se exporta el programa servidor al archivo *servidor.jar*. Como éste requiere la librería de JDBC para su funcionamiento se necesita disponer del conector de dicha librería (disponible junto a la documentación del Proyecto) junto al archivo recién generado. A la hora de ejecutar el servidor habrá que añadir ambos al classpath.

5. EJECUCIÓN DEL SERVIDOR

En este punto, el servidor puede ya arrancar para quedarse a la espera de recibir notificaciones de los clientes que instalen el plugin para *Vuze*. La ejecución del servidor se debe realizar desde la consola del sistema situándose en la carpeta donde se encuentren ambos archivos .jar, el del conector JDBC y el del programa servidor, lanzando el siguiente comando:

```
>> java –classpath servidor.jar;mysql-connector-java-5.1.13-
bin.jar servidor.PluginServidor
```

La ejecución del servidor habrá sido efectuada correctamente si aparecen en la consola las siguientes líneas:

```
-- Servidor configurado --
-- Servidor conectado a la base de datos --
Esperando cliente...
```

ANEXO 3. Manual del plugin para el usuario

REQUISITOS PREVIOS

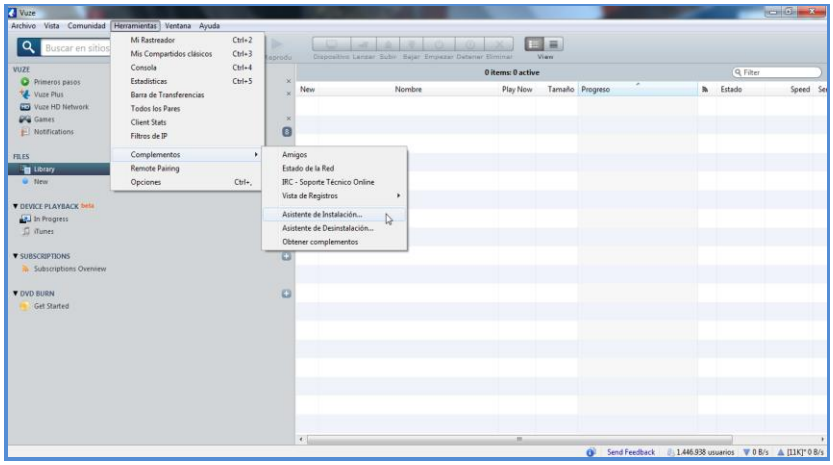
- ✓ Disponer del JRE (*Java Runtime Environment*). Si no es así, se puede descargar la versión para la plataforma de la máquina donde se quiera instalar en la sección JRE de la página de Descargas de Oracle:
  http://www.oracle.com/technetwork/java/javase/downloads/
  Para la ayuda en la instalación en cada plataforma se dispone de guías en:
  http://www.java.com/es/download/help/index_installing.xml

- ✓ Descargar e instalar *Vuze* según el Manual de Instalación disponible en:
  http://azureus.com.es/manuales/manual-de-instalacian-de-azureus

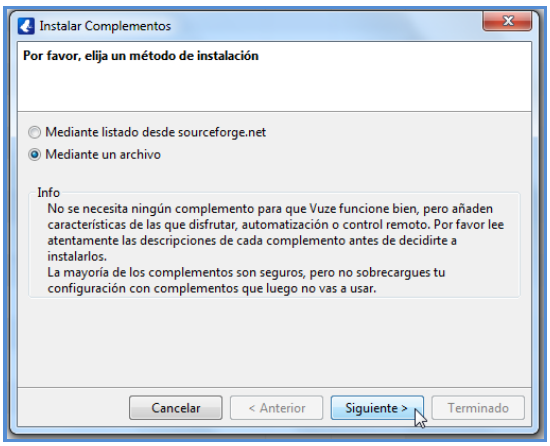
- ✓ Disponer del archivo *plugin_vuze_0.1.jar* que contiene el plugin.

INSTALACIÓN

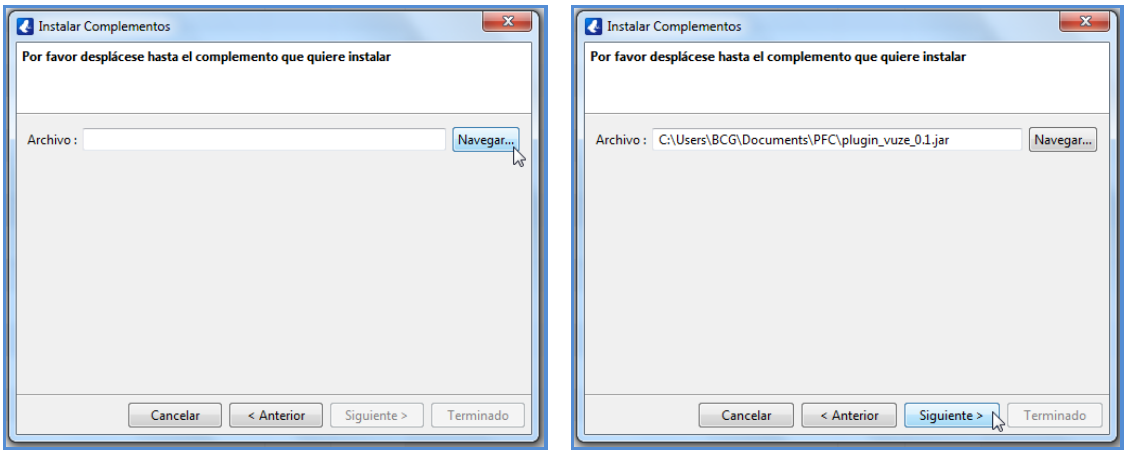
1. Abrir *Vuze* y acceder a la pestaña *Herramientas* y después marcar la opción *Complementos*. Tras ello seleccionar la opción *Asistente de Instalación…*



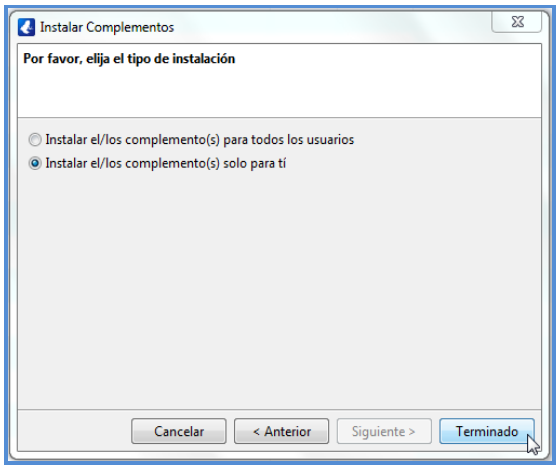
2. Elegir el método de instalación *Mediante un archivo* y pulsar *Siguiente*.



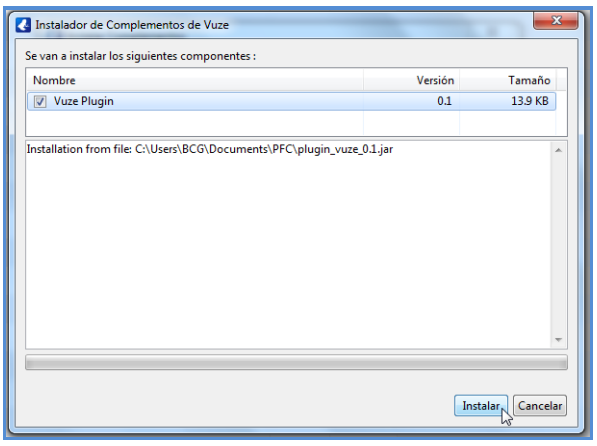
3. Pulsar en el botón *Navegar* para acceder al directorio del ordenador y localizar en los directorios el archivo .jar que contiene el plugin. Haciendo click en *Abrir* se selecciona dicho archivo. Pulsar *Siguiente* para continuar.



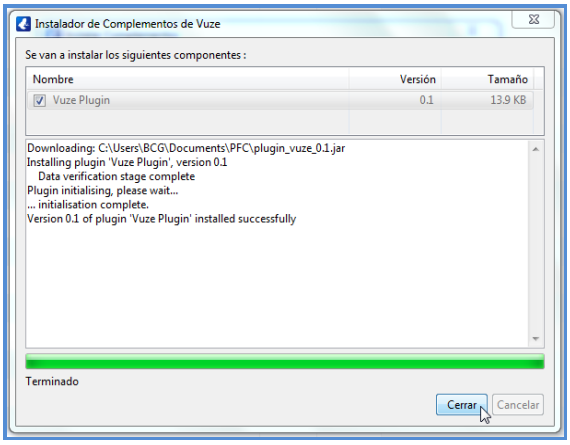
4. Elegir el tipo de instalación *Instalar el/los complemento(s) sólo para ti* y pulsar *Terminado*.



5. Confirmar la instalación del plugin indicado pulsando en *Instalar*.



6. Esperar a que el Instalador de Complementos complete la instalación. Esto habrá ocurrido cuando la ventana muestre la siguiente apariencia con la barra de progreso completamente verde y se indique el mensaje Terminado.



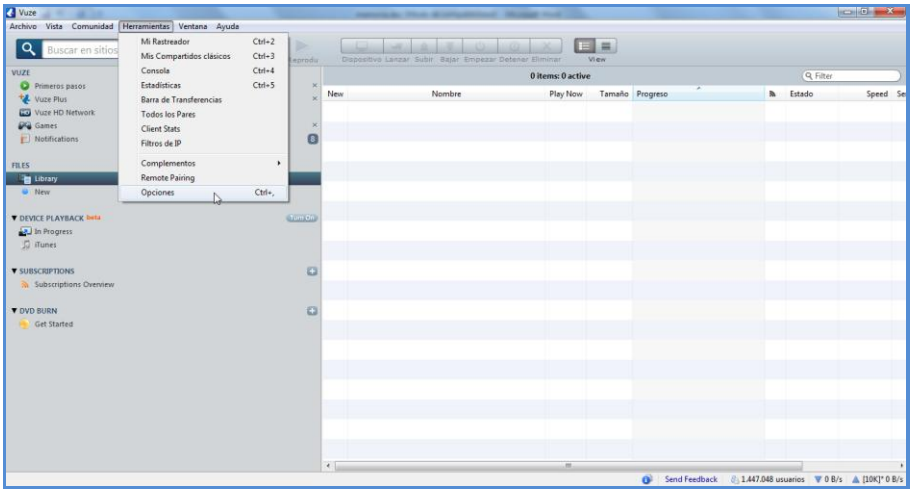
Tras completar la instalación es necesario pulsar en *Cerrar* de la ventana del Instalador de Complementos para regresar a la pantalla principal de *Vuze*.

7. Instalación del plugin completa.

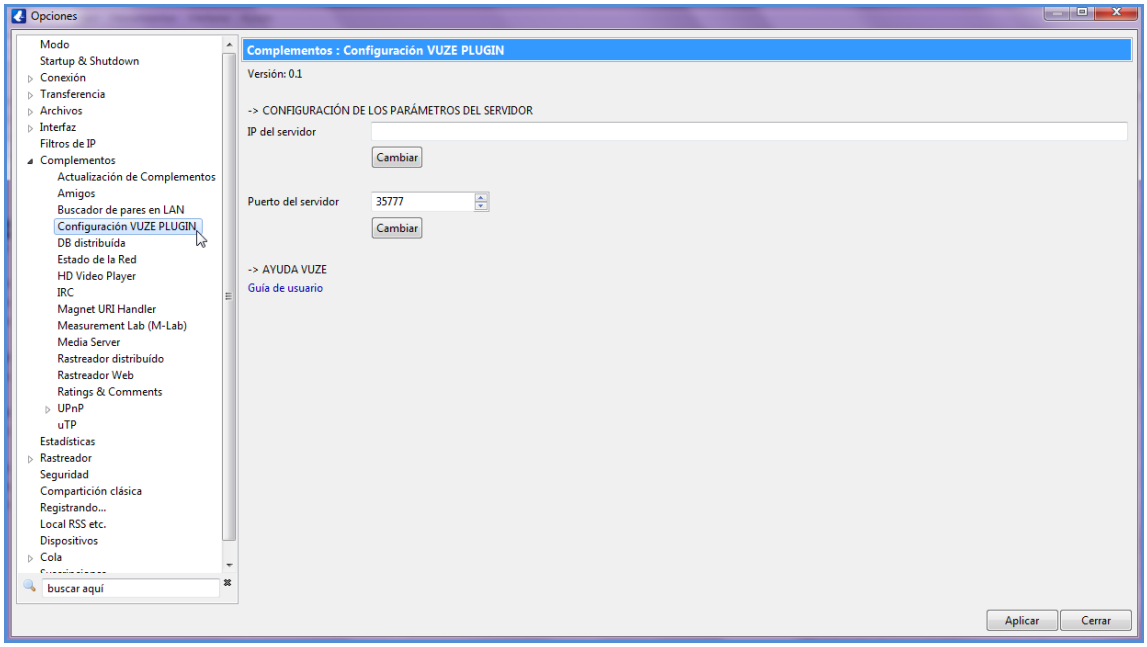
ACCESO A LA PÁGINA DE CONFIGURACIÓN

Para poder visualizar la página de configuración donde modificar los parámetros de configuración del servidor que atiende el plugin se deben seguir las siguientes instrucciones:

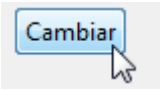
1. Pulsar en el menú la pestaña *Herramientas* y seleccionar *Opciones*.



2. En la parte izquierda de la ventana de Opciones abrir la pestaña *Complementos* y pulsar sobre *Configuración VUZE PLUGIN*.



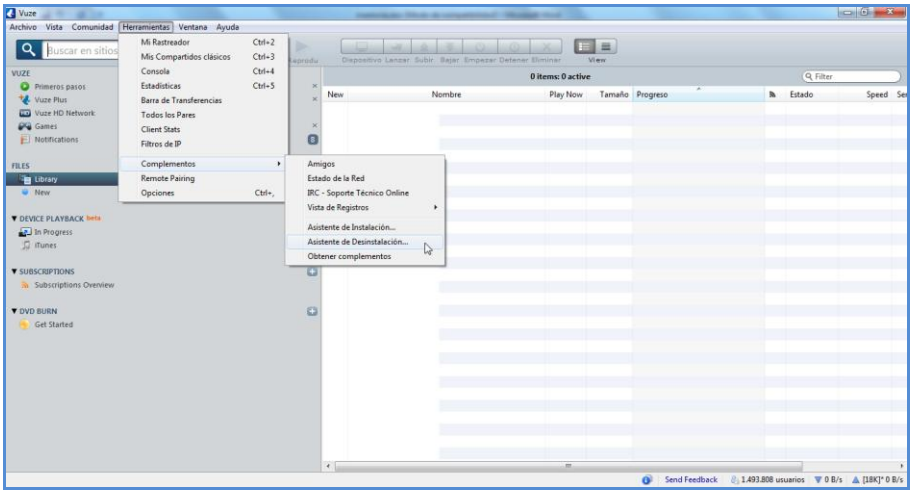
3. Realizar los ajustes necesarios en la configuración del plugin pulsando en el botón *Cambiar* para aceptar los cambios.



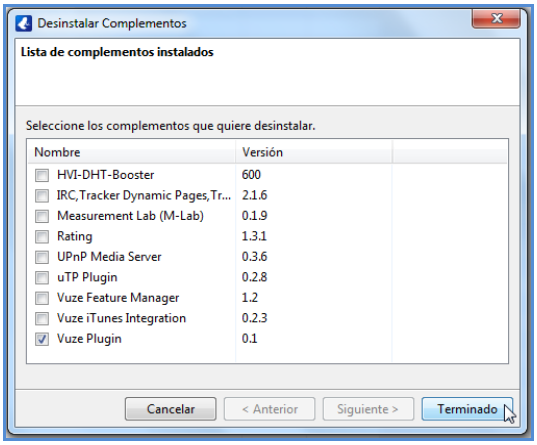
4. Pulsar en *Aplicar* para aplicar los cambios y en *Cerrar* para volver a la pantalla principal de *Vuze*.

DESINSTALACIÓN

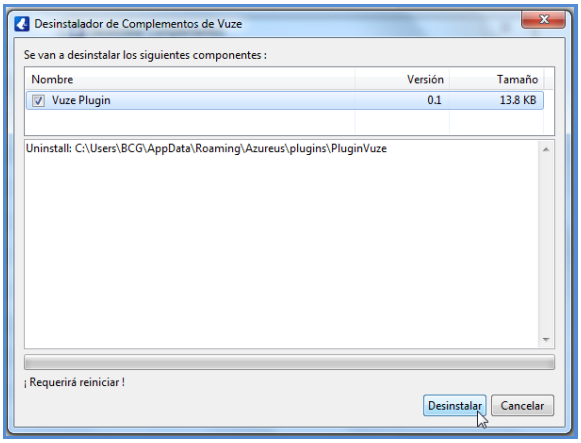
1. Entrar en la opción *Herramientas* de la barra de Herramientas de *Vuze* y pulsar sobre la opción *Complementos*. Tras ello seleccionar la opción *Asistente de Desinstalación…*



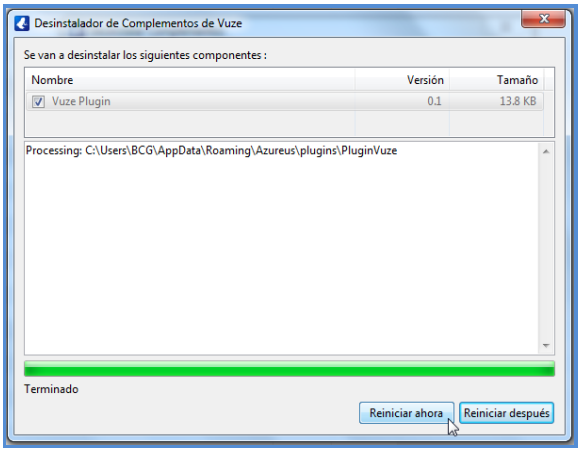
2. Seleccionar en la Lista de Complementos Instalados el plugin a desinstalar y pulsar en *Terminado.*



3. Confirmar la acción pulsando sobre *Desinstalar* y esperar a que se complete la desinstalación.



4. Cuando aparezca el mensaje *Terminado* hacer click en *Reiniciar Ahora* para reabrir *Vuze*.



5. Desinstalación del plugin completa.